НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

Інститут прикладного системного аналізу

Кафедра математичних методів системного аналізу

До захисту допущено:

В.о.завідувача кафедри

_____ Оксана ТИМОЩУК

«___»_____20__ р.

Дипломна робота

на здобуття ступеня бакалавра

за освітньо-професійною програмою «Системний аналіз і управління»

спеціальності 124 «Системний аналіз»

на тему: «Розв'язання задачі заповнення пропусків даних альтернативними методами»

Виконав:
студент IV курсу, групи КА-64
Маркін Іван Дмитрович _____

Керівник:
асистент кафедри ММСА,
Кухарєв С.О. _____

Консультант з економічного розділу:
доцент, к.е.н. Шевчук О.А.

Консультант з нормконтролю:
доцент, к.т.н. Коваленко А.Є. _____

Рецензент:
доцент кафедри СП
Безносик О.Ю. _____

Засвідчую, що у цій дипломній роботі немає запозичень з праць інших авторів без відповідних посилань.
Студент _____

Київ – 2020 року

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Інститут прикладного системного аналізу
Кафедра математичних методів системного аналізу

Рівень вищої освіти – перший (бакалаврський)

Спеціальність – 124 «Системний аналіз»

Освітньо-професійна програма «Системний аналіз і управління»

ЗАТВЕРДЖУЮ
В.о.завідувача кафедри
_____ Оксана ТИМОЩУК
«___»_____20__ р.

ЗАВДАННЯ
на дипломну роботу студенту
Маркін Іван Дмитрович

1. Тема роботи «Розв'язання задачі заповнення пропусків даних альтернативними методами», керівник роботи Кухарєв Сергій Олександрович, асистент кафедри ММСА, затверджені наказом по університету від «_25_»_травня_ 20_20_ р. №_1143-с__

2. Термін подання студентом роботи 08 травня 2020 року _____

3. Вихідні дані до роботи

1. Операційна система Windows 10
2. Частота процесора 2.8 ГГц
3. Мова програмування Python
4. Середовище розробки – PyCharm Community Edition 2019.2.1
5. Бібліотеки що використовувалися: Pandas, Numpy, Matplotlib, Sklearn, Scipy, Time, Sys.

4. Зміст роботи

1. Проаналізувати існуючі методи заповнення пропусків даних
2. Проаналізувати необхідну теорію для реалізації існуючих методів

3. Розробити програмний продукт для моделювання нейронних мереж з довгою короткостроковою пам’яттю
4. Розробити систему оцінки побудованої моделі
5. Виконати економічний аналіз програмного продукту

5. Перелік ілюстративного матеріалу (із зазначенням плакатів, презентацій тощо)

1. Перзентація

6. Консультанти розділів роботи*

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Економічний	Шевчук О.А., доцент	21.04.20	28.05.20

7. Дата видачі завдання _____

Календарний план

№ з/п	Назва етапів виконання дипломної роботи	Термін виконання етапів роботи	Примітка
1	Отримання завдання	10.04.20	
2	Аналіз існуючих методів	19.04.20	
3	Вибір алгоритмів для розв’язання задачі. Пошук вхідних даних	29.04.20	
4	Розробка програмного продукту	13.05.20	
5	Тестування програмного продукту. Аналіз результатів	15.05.20	
6	Функціонально-вартісний аналіз програмного продукту	17.05.20	
7	Оформлення дипломної роботи	20.05.20	

Студент Іван Дмитрович МАРКІН

_____

Керівник Сергій Олександрович КУХАРЄВ

РЕФЕРАТ

Дипломна робота: 68 с., 8 табл., 8 рис., 2 дод., 13 джерел.

РОЗВЯЗАННЯ ЗАДАЧІ ЗАПОВНЕННЯ ПРОПУСКІВ ДАНИХ АЛЬТЕРНАТИВНИМИ МЕТОДАМИ

Об’єкт дослідження: Задача заповнення пропусків даних.

Предмет дослідження: методи вирішення задачі заповнення пропусків даних.

Мета роботи: проаналізувати особливості задачі заповнення пропусків даних, провести порівняльний аналіз існуючих методів заповнення пропусків даних.

Методи дослідження: чисельні експерименти з порівнянням результатів, що реалізовані за допомогою мови програмування Python.

У роботі розглядається задача заповнення пропусків даних. Існує багато методів вирішення даної задачі, які різняться за своєю складністю в реалізації. Цілю цієї роботи є порівняльний аналіз існуючих методів заповнення пропусків даних.

ABSTRACT

Bachelor's thesis: 68 p., 8 tab., 8 fig., 2 add., 13 references.

ALTERNATIVE METHODS OF FILLING DATA GAPS

Object of research: The task of filling data gaps.

Subject of research: methods of solving the problem of filling data gaps.

The purpose of the work: to analyze the features of solving the problem filling data gaps, to compare existing methods of filling data gaps.

Research methods: numerical experiments comparing the results implemented using the Python programming language.

The paper considers the problem of filling data gaps. There are many methods for solving this problem, which differ in their complexity of implementation. The main task of this work is to compare existing methods of filling data gaps.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ

MCAR - Missing Completely At Random.

MAR - Missing At Random.

MNAR - Missing Not At Random.

ПП – програмний продукт.

ВСТУП

У практичних завданнях аналізу даних вибірки часто містять в собі пропущені значення. Причини можуть бути різними, наприклад, відсутність відповіді респондента на конкретне запитання анкети, відмова датчика для вимірювань показника, помилки в програмному забезпеченні під час запису даних. Часто викиди даних також можна розглядати як пропуски. До викидів можна віднести данні, які явно суперечать даним з усієї вибірки.

За рідкісним винятком алгоритми машинного навчання не працюють з вибірками, що мають пропущені значення. Тому виникає необхідність у процедурі заповнення даних – процедурі попередньої обробки. Існують різні підходи до вирішення даного завдання, які різняться за своєю природою, областю застосування і обчислювальною складністю.

У даній роботі описані основні методи відновлення пропусків даних, проведено їх порівняння на декількох наборах даних.

РОЗДІЛ 1 ЗАДАЧІ ЗАПОВНЕННЯ ПРОПУСКІВ ДАНИХ

1.1 Джерела та види пропусків даних

Часто в даних з якими необхідно працювати аналітикам при аналізі та моделюванні досліджуваних процесів наявні пропуски. Перед початком розв'язування задачі заповнення пропусків даних необхідно виявити механізм формування пропусків.

Розрізняють 3 основних механізми формування пропусків в даних: MCAR, MAR, MNAR.

MCAR (Missing Completely At Random) – механізм рівномірного формування пропусків, тобто ймовірність пропуску для кожного запису однакова. Ймовірність $P(x_j \text{пропущено})$ не залежить ні від інших x ні від самого x_j, іншими словами $P(x_j \text{пропущено})$- постійна для всіх спостережень. Прикладом MCAR, при зборі даних може слугувати використання приладів, які працюють на батарейках. В такому випадку пропуски в даних повністю залежать від вдачі. Іншим прикладом MCAR є випадкова вибірка групи населення, де кожен член має однаковий шанс потрапити у вибірку. Члени популяції які не брали участі в опитування і є MCAR.

MAR (Missing At Random) – ймовірність пропуску може бути обрахована в залежності від іншої наявної в даних інформації. Ймовірність $P(x_j \text{пропущено})$ не залежить від x_j, але може залежати від інших x. На практиці дані зазвичай пропущені не випадково, для них існує певна закономірність. Люди, які займають керівні посади і / або які отримали вищу освіту частіше, ніж інші респонденти, не відповідають на питання про свої доходи. Оскільки посада і освіта сильно корелюють з доходами, то в такому випадку пропуски в графі доходи вже не можна вважати абсолютно випадковими, тобто говорити про випадок MCAR не представляється

можливим. Важливо зазначити що механізм MAR частіше зустрічається на практиці ніж MCAR.

MNAR (Missing Not At Random) - механізм формування пропусків, при якому дані відсутні в залежності від невідомих чинників. MNAR передбачає, що ймовірність пропуску могла б бути описана на основі інших атрибутів, але інформація по цим атрибутам в наборі даних відсутня. Як наслідок, ймовірність пропуску неможливо виразити на основі інформації, що міститься в наборі даних. Відомий приклад MNAR з області медичних досліджень полягає в тому що респондент з більшою ймовірністю не братиме участь в опитувані, якщо лікування призводить до дискомфорту. Такі пропуски даних не є випадковими, а тому мають бути змодельовані, інакше дослідник повинен прийняти деякі упередженості в своїх висновках.

1.2 Класифікація методів вирішення задачі заповнення пропусків даних

Різноманітні ситуації та причини виникнення пропусків в даних призвело до появи багатьох досліджень в даній області. Велика кількість методів розв’язування задачі заповнення пропусків даних вимагає систематизації підходів та класифікації методів [Kalton, Kasprzyk]. У вказаній вище праці приведені основні принципи методів відновлення даних. Таким чином більшість розроблених методів підпадають під наведену схему класифікації (рис.1.1).

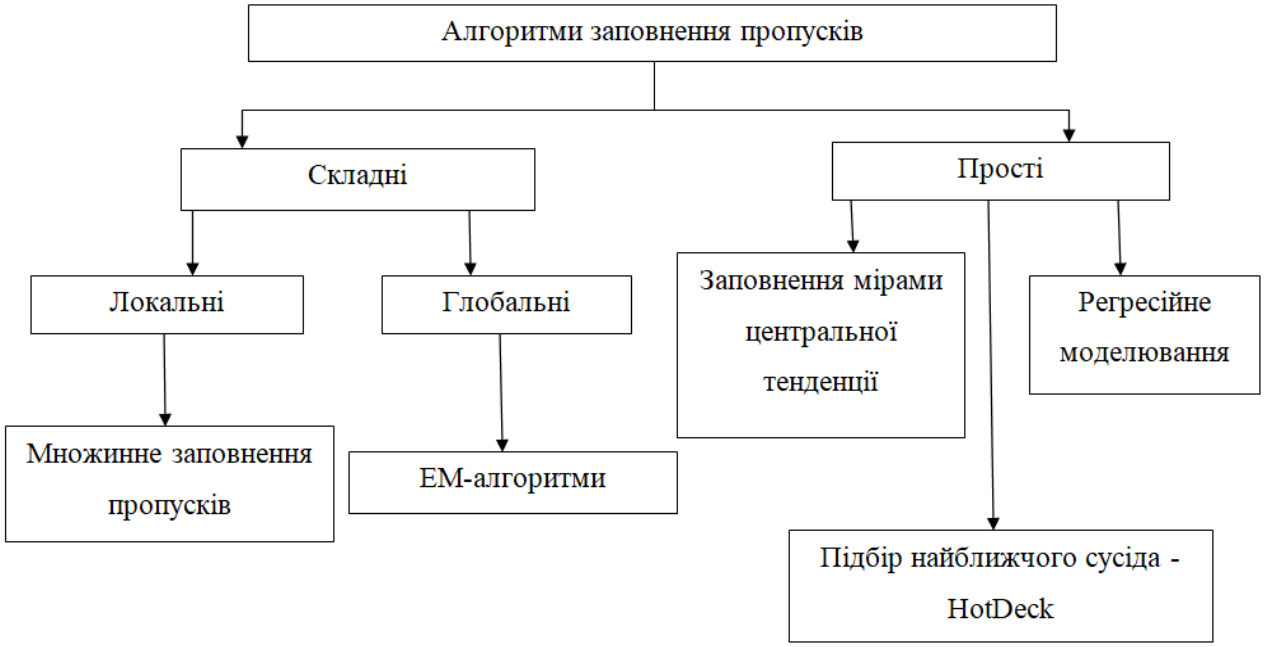


Рисунок 1.1 - Класифікація алгоритмів заповнення даних

Прості алгоритми – засновані на простих арифметичних операціях, відстані між об’єктами, регресійним моделюванням.

Складні алгоритми – ітеративні алгоритми. Даний тип алгоритмів передбачає оптимізацію деякого функціонала, який в свою чергу обчислює точність підставлених значень. Складні алгоритми можна розділити на глобальні та локальні.

Глобальні алгоритми при передбачені кожного пропущеного значення використовують усі об’єкти вибірки.

Локальні алгоритми при передбачені кожного пропущеного значення використовують об’єкти які знаходяться в певному околі передбачуваного значення.

Можна стверджувати, що теорія відновлення пропусків даних постійно розвивається, з’являються нові алгоритми та удосконалюються існуючі. Це пов’язано з тим що не існує алгоритму який був би прийнятний та давав кращі результати в абсолютній більшості випадків.

Заповнення мірами центральної тенденції (середнім по всій вибірці або середніми по групах) - застосування має сенс тільки в разі проходження даних

умові MAR, дану групу методів легко можна реалізувати; недоліки - спотворення розподілу даних, зменшення дисперсії.

Заповнення по регресії. В основу даної групи методів покладені добре відомі алгоритми регресійного аналізу [Двоенко]. З умов застосування даного методу можна виділити вимогу про приналежність даних умові MAR (хоча для окремих випадків можливе застосування більш слабких вимог) і вимоги, які стосуються виконання передумов регресійного аналізу. Недоліки подібних методів очевидні: якість передбачення (відновлення пропусків) безпосередньо залежить від успішного вибору взятої за основу регресійній моделі.

Метод заміни пропущеного значення середнім з найближчих присутніх елементів змінної. Даний метод є ефективним розвитком методу заміни пропусків загальним середнім, і експерименти показують гарну точність методу в разі одиночних пропусків на досить гладких рядах даних. Завдяки простоті реалізації можна навіть рекомендувати використання даного методу в наведених вище умовах, але тільки в них. Наявність в даних групових пропусків або сильні флуктуації значень ряду зводять ефективність методу до нуля.

Методи множинного заповнення. Основна їхня перевага в тому, що вони долають недолік методів однократного заповнення в сенсі значного розкиду дисперсії оцінки.

EM-алгоритм - відноситься до категорії методів моделювання [Загоруйко, Йолкіна]. Особливість цих методів - побудова моделі породження пропусків з подальшим отриманням висновків на підставі функції правдоподібності, побудованої за умови справедливості даної моделі, з оцінюванням параметрів методами типу максимальної правдоподібності. Відзначимо, що якщо інші методи відновлення пропусків вимагають, щоб дані відповідали умові MAR (або MCAR як жорсткішого), то для даних методів можлива побудова моделей, що враховують конкретну специфіку області, як наслідок, можлива постановка слабших умов до даних. Недолік - необхідність побудови моделі породження пропусків.

1.3 Висновки до розділу

При вирішені практичних задач часто виникає потреба попередньої обробки даних. Заповнення пропусків даних є однією з задач попередньої обробки даних без вирішення якої неможлива подальша обробка даних. Різноманіття джерел та видів пропусків даних призвело до появи великої кількості алгоритмів заповнення пропусків даних. Теорія відновлення пропусків даних постійно розвивається, з’являються нові методи та модифікуються старі.

У даному розділі було описано основні механізми формування пропусків даних.

Була сформована класифікація методів відновлення пропусків даних.

РОЗДІЛ 2 МЕТОДИ ЗАПОВНЕННЯ ПРОПУСКІВ ДАНИХ

2.1 Постановка задачі

Є матриця об'єктів-ознак $X^{n\times d}$, n - кількість об'єктів, d - кількість ознак. Частина значень матриці пропущені. Необхідно отримати матрицю об'єктів-ознак без пропущених значень з метою подальшого застосування алгоритмів машинного навчання.

2.2 Методи відновлення даних

В цьому пункті описано основні методи заповнення пропусків даних, наведено їх математичне обґрунтування.

2.2.1 Ігнорування об'єктів з пропущеними значеннями

Найпростішим засобом для вирішення проблеми пропущених значень у вибірці є ігнорування об'єктів, що мають пропуски. Ігнорування об’єктів – простий алгоритм за якого об’єкти, які мають в своїх показниках пропуски даних, не приймають участі у побудові моделей на основі даного набору даних. Такий метод можна застосовувати тільки в тому випадку, коли мала частина об'єктів вибірки має пропущені значення (<5%). В іншому випадку є ризик втрати великої кількості даних, що зробить неможливим подальший аналіз даних та побудови моделей на їх основі. Також цей метод може привести до значного зсуву при оцінці параметрів. Однак, відомо, якщо виконується гіпотеза про цілком випадковий механізм формування пропусків MCAR, метод ігнорування пропущених значень призводить до незміщених оцінок параметрів та прийнятних результатам. Перевагою даного підходу є простота і неможливість зіпсувати дані шляхом заміни пропусків. У разі досить великого

розміру вибірки та незначної кількості пропусків метод може показувати гарні результати. Альтернативним варіантом в разі наявності пропусків в невеликій кількості ознак є видалення таких ознак з вибірки.

2.2.2 Заміна спеціальним значенням

Іншим найпростішим методом є заміна пропусків на спеціальне заздалегідь відоме значення таке, як, наприклад, 0 або -1. Даний підхід дозволяє не зменшувати розмір вибірки, однак може вносити значення, які сильно відрізняються від заданих.

Для подальшого застосування методів, заснованих на деревах, розумно заповнювати пропуски за допомогою значення, яке не зустрічається у вибірці, наприклад, -1 для невід'ємних значень ознак. Для методів, чутливих до масштабу ознак, пропуск замінюється за допомогою 0.

2.2.3 Заміна найчастішим або середнім значенням

Ще одним простим методом відновлення пропусків є заміна на моду або середнє значення за конкретною ознакою. У разі категоріальної ознаки усі пропуски заміняються на значення, яке найбільш часто зустрічається, в разі кількісної ознаки - на середнє значення за ознакою. Даний метод, на відміну від двох попередніх, враховує наявні дані і усереднює їх. Перевагою підходу є простота, однак на практиці виникає проблема у визначенні, чи є конкретна ознака категоріальною або кількісною, особливо при їх великій кількості. Даний метод також бажано не використовувати в вибірках які не є цілком випадковими, а також якщо кількість пропущених даних залежить від змінних. Ще одним недоліком методу заміни середнім або модою є те, що він не додає нової інформації та призводить до заниження помилок.

2.2.4 Заміна за допомогою сингулярного розкладу

У машинному навчанні сингулярний розклад [Imuting Missing Data for Gene Expression Arrays] використовується для наближення матриці матрицею меншого рангу за наступною схемою:

Знаходиться сингулярний розкладання матриці:

$$X = U\Sigma V^*,$$

де $U^{n\times n}, V^{d\times d}$ −відповідно права та ліва ортогональні матриці

$\Sigma^{n\times d}$ − діагональна матриця з власними числами, яка має вигляд:

$$\Sigma = \begin{pmatrix} \lambda_1 & 0 & \dots & 0 \\ 0 & \lambda_2 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \lambda_n \end{pmatrix}$$

де $\lambda_1 > \lambda_2 > \cdots > \lambda_n$- власні числа.

Знайдемо матрицю Σ' з Σ шляхом занулення всіх діагональних елементів крім k найбільших, тобто $\lambda_i = 0$, для $i > k$.

Матриця Σ' тепер має вигляд:

$$\Sigma' = \begin{pmatrix} \lambda_1 & \dots & 0 & 0 & \dots & 0 \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ 0 & \dots & \lambda_1 & 0 & \dots & 0 \\ 0 & \dots & 0 & 0 & \dots & 0 \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ 0 & \dots & 0 & 0 & \dots & 0 \end{pmatrix}$$

Відновлення матриці:

$$X_{approx} = U\Sigma' V^*,$$

В даному методі сингулярний розклад застосовується спочатку для скорочення розмірності матриці, після чого пропущені значення замінюються по відновленій з сингулярного розкладу матриці меншого рангу.

Для початкової ініціалізації використовується заміна пропущених значень середнім значенням ознаки. Далі ітеративно до матриці об'єктів і ознак застосовується сингулярний розклад з відкиданням найменших власних чисел,

після чого пропущені значення замінюються по відновленій матриці до збіжності або досягнення максимального заданого числа ітерацій. Далі наведено приклад псевдокоду алгоритму.

Algorithm SVD Imputer

X [missing] simple initialize (X)

for iteration = 1 to max_iterations do

U, Σ, V ← SV D (X)

Σ′ ← reduce (Σ, k)

$X_{approx} \leftarrow U\Sigma'V^*$

X [missing] ← X_{approx}

end for

2.2.5 Заміна за допомогою методу найближчих сусідів

З припущення про те, що близькі об'єкти за значеннями серед заповнених ознак близькі в ознаках, значення яких може бути пропущено (гіпотеза компактності), виникає застосування методу k найближчих сусідів для відновлення пропусків у даних. Реалізація аналогічна використанню класичного методу k найближчих сусідів за винятком того, передбачається відразу кілька пропущених ознак для кожного об'єкта.

Для кожного об’єкта з пропущеними даними, знаходяться k найближчих сусідів, для цього задається формула пошуку відстані між об’єктами. В даній роботі можливе використання двох на вибір формул:

$$p(x, x') = \sqrt{(x - x')^2}$$

та

$$p(x, x') = |x - x'|,$$

Знаходяться $x_1, \dots, x_k$, для яких

$$p(x_1, x') < p(x_2, x') < \cdots < p(x_k, x') < \cdots$$

Значення шуканого об’єкта дорівнює середньому з k знайдених ближчих сусідів

$$x' = \frac{x_1 + x_2 + \cdots + x_k}{k}$$

Псевдокод алгоритму наведено нижче.

Algorithm kNN Imputer

X_{full} = X [rows without missing values]

for row with missing values do

$X_{neighbors}$ ← find k neighbors (row, X_{full}, k)

row [missing] ← mean ($X_{neighbors}$)

end for

Описаний в вище алгоритм передбачає достатню кількість об'єктів без пропущених значень, за якими відбувається відновлення. Якщо більшість об'єктів має пропуски можна виконувати початкову ініціалізацію пропущених значень і за X_{full} брати всю матрицю об'єктів-ознак.

Метод k найближчих сусідів може бути модифіковано шляхом додання спеціальних коефіцієнтів для атрибутів. Деякі атрибути можуть бути важливішими інших, тому кожному атрибуту може бути задана певна вага задана експертом.

2.2.6 Заміна за допомогою випадкового лісу

Замінити пропущені значення в конкретній ознаці можна, передбачивши його за іншими ознаками за допомогою одного з алгоритмів машинного навчання. Так як серед ознак, за якими проводиться навчання, є пропущені значення, то необхідно їх спочатку замінити за допомогою одного з найпростіших методів відновлення пропусків.

В даному випадку ініціалізацію пропущених значень проводитися за допомогою заміни середнім значенням за ознакою, як алгоритм для уточнення

пропусків використовується випадковий ліс. Для кожної ознаки, що має пропущені значення, проводиться навчання по об'єктах, які не мають пропусків в даній ознаці, для решти об'єктів проводиться заміна значень даної ознаки за допомогою навченого алгоритму. Ця процедура повторюється протягом декількох ітерацій до збіжності або до максимального встановленого числа ітерацій.

Для передбачення ознаки, значення якої відсутнє алгоритм випадкового лісу передбачає побудову певної кількості дерев, які мають структуру подібну то тої що зображена на рисунку 2.1.

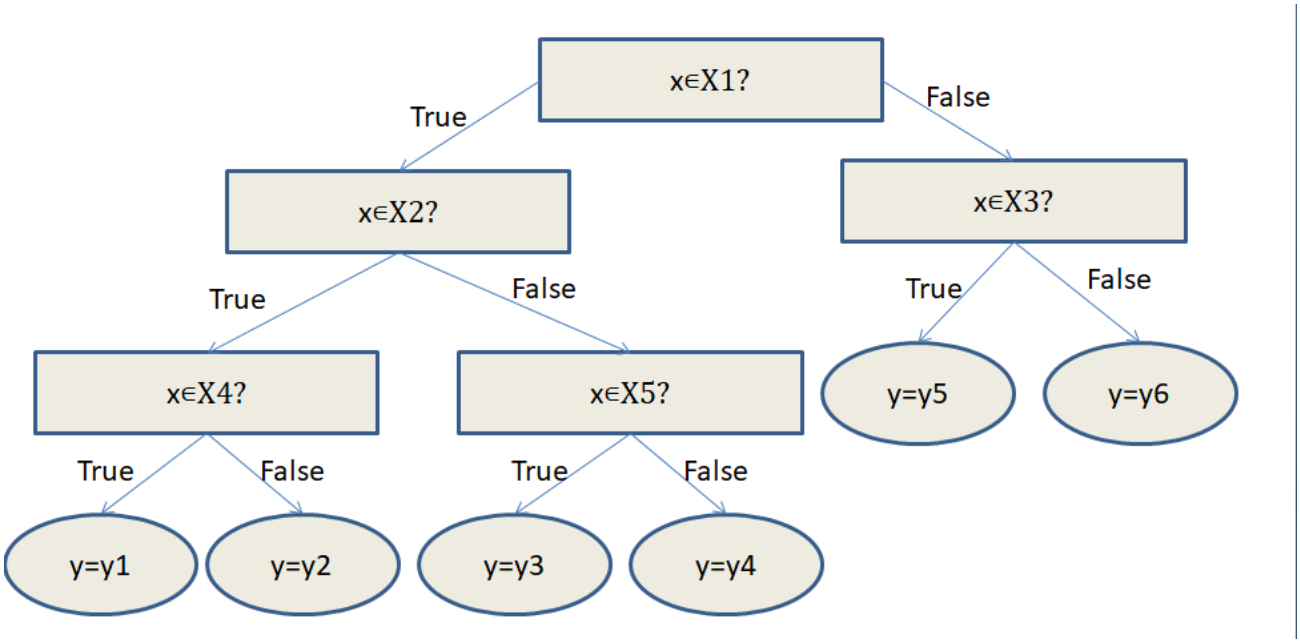


Рисунок 2.1 Структура дерева

У цій структурі x – відома ознака об’єкта, y – відсутня ознака, X_i −певна множина до якої може потрапити x.

Псевдокод алгоритму наведено нижче.

Algorithm 3 RF Imputer

X [missing] ← simple initialize (X)

for iteration = 1 to max_iterations do

for column with missing values do

X_{train} ← X [without missing values]

X_{test} ← X [with missing values]

X [missing, column] ← predict RF (X_{train}, X_{test})

end for

end for

2.2.7 Заміна за допомогою лінійної регресії

Використовується стратегія, аналогічна попередньому методу, за винятком того, що замість випадкового лісу використовується лінійна регресія. Псевдокод даного методу аналогічний псевдокоду випадкового лісу.

2.2.8 Заміна за допомогою EM-алгоритму

EM-алгоритм - відноситься до категорії методів моделювання. Особливість цих методів - побудова моделі породження пропусків з подальшим отриманням висновків на підставі функції правдоподібності, побудованої за умови справедливості даної моделі, з оцінюванням параметрів методами максимальної правдоподібності. Відзначимо, що якщо інші методи відновлення пропусків вимагають, щоб дані відповідали умові MAR (або MCAR як жорсткішого), то для даних методів можлива побудова моделей, що враховують конкретну специфіку області, як наслідок, можлива постановка слабших умов до даних. Недолік - необхідність побудови моделі породження пропусків.

При вирішенні задачі розділення суміші, використовуючи принцип максимальної правдоподібності «у лоб», призводить до занадто громіздкої оптимізаційної задачі. Цю проблему й вирішує EM- алгоритм.

EM-алгоритм складається з ітераційного повтору двох кроків: E та M кроки. На E-кроці обчислюється очікуване значення (expectation) вектора прихованих змінних G за поточним наближенню вектора параметрів Θ. На M-кроці вирішується завдання максимізації правдоподібності (maximization) і знаходиться наступне наближення вектора Θ за поточними значеннями векторів G і Θ.

Маючи інформацію про розподіл даних у вибірці, можна відновити пропущені значення. Використовується ЕМ-алгоритм [Dempster, Laird, Rubin],[Воронцов] для відновлення параметрів суміші нормальних розподілів.:

$$p(x) = \sum_{k=1}^{K} \pi_k N(\mu_k, \Sigma_k), \pi_k \geq 0, \sum_{k=1}^{K} \pi_k = 1.$$

На Е-кроці знаходимо коефіцієнти:

$$g_{mk} = \frac{\pi_k p_k(X^m)}{\pi_1 p_1(X^m) + \pi_2 p_2(X^m) + \cdots + \pi_K p_K(X^m)}$$

де $p_k(X)$ – платність багатовимірного нормального розподілу.

$$p_k(X^m) = \frac{1}{\Sigma_k (2\pi)^{N/2}} \exp\left\{-\frac{\sum_{n=1}^{N}(X_n^m - \mu_{kn})^2}{2\Sigma_k^2}\right\}$$

На М-кроці оновлюємо коефіцієнти

$$\pi_k = \frac{1}{M}\sum_{m=1}^{M} g_{mk}$$

$$\mu_{kn} = \frac{1}{M\pi_k}\sum_{m=1}^{M} g_{mk} X_n^m$$

$$\sigma_{kn}^2 = \frac{1}{M\pi_k}\sum_{m=1}^{M} g_{mk}(X_n^m - \mu_{kn})^2$$

За середнім значенням і матрицями коваріації ознак обчислюються коефіцієнти регресії:

$$\beta_k = cov\,(X_{-j}, X_j)\,\Sigma_k^{-1},$$

де X_j - відновлювана ознака,

X_{-j}- матриця без відновлюваної ознаки.

Передбачається, що середні значення ознак X дорівнюють нулю. За отриманими коефіцієнт обчислюються значення на місцях пропусків за іншими ознаками цього об'єкта:

$$X_j^k = X_{-j}\beta_k^T$$

Підсумкове передбачення пропущеного значення усереднюється за розподілом:

$$X_j = \sum_{k=1}^{K} \pi_k X_j^k.$$

В якості початкової ініціалізації пропущених значень використовується заповнення середнім значенням кожної ознаки. Далі ітеративно відновлюються параметри розподілу і уточнюються пропущені значення до збіжності або досягнення максимального заданого числа ітерацій. Псевдокод алгоритму наведено нижче.

Algorithm EM Imputer

X [missing] ← simple initialize (X)

for iteration = 1 to max_iterations do

π, μ, Σ ← X

for row with missing values do

for k = 1 to K do

$coef_k$ ← calculate (μ_k, Σ_k)

$predict_k$ ← regression($coef_k$, X [row, nonmissing])

end for

X [row, missing] ← $\sum_{k=1}^{K} predict_k$

end for

end for

2.2.9 Заміна за допомогою методу k середніх

Аналогічно методу k найближчих сусідів передбачається, що близькі за одними ознаками об'єкти повинні бути близькі і за іншими ознаками. Однак на відміну від методу k найближчих сусідів шукають не найближчі сусіди для кожного об'єкта з пропущеними значеннями, а використовується інформація про центр кластера, в який потрапив конкретний об'єкт з пропусками.

Зауважимо, що для розбиття на кластери необхідна початкова ініціалізація пропущених значень.

В даному випадку виконується ініціалізація пропущених значень за допомогою заміни середнім значенням за ознакою, кластеризація проводиться методом k середніх [MacQueen]. Пропущені значення замінюються на відповідні їм значення центру кластера, в який потрапив кожен об'єкт з пропусками. Дана процедура проводиться протягом декількох ітерацій до збіжності або по досягненню максимального заданого числа ітерацій. Псевдокод алгоритму наведено нижче.

Algorithm K-means Imputer
X [missing] ← simple initialize (X)
for iteration = 1 to max_iterations do
    centroids ← kmeans (X)
    X [missing] ← centroids
end for

2.2.10 Алгоритм ZET

Алгоритм ZET запропонований Загоруйко Н.Г. [Прикладные методы анализа данных и знаний]. В його основі лежать три припущення (гіпотези):

Гіпотеза надмірності – передбачається, що в таблиці має місце надмірність в рядках (об’єкти можуть бути схожими одне на одного) та в стовбцях (між властивостями об’єктів може бути певна залежність)

Гіпотеза аналогічності – передбачається що якщо два об’єкта «схожі» за значенням (n-1) властивістю то вони будуть схожі і за n властивістю.

Гіпотеза компактності, суть якої описана нижче.

Суть алгоритму Zet полягає в підборі для кожного пропуску значення не з усієї сукупності повних спостережень, а з певної її частини, так званої компонентної матриці. Вона складається з компонентних рядків і стовпців.

Компонентність деякого рядка або об'єкта являє собою величину обернено пропорційну декартовій відстані до цільового рядка (неповного спостереження з пропуском) в просторі, осі якого задані змінними - розглянутими характеристиками об'єктів.

За даними компонентної матриці потім будується функціональна залежність прогнозованого значення від відповідного значення в компетентної матриці, на основі якої, потім, прогнозується значення пропуску.

Відновлення пропущених значень відбувається наступним чином: для кожного пропущеного значення в нормалізованих даних обчислюються компетентності всіх рядків, які не мають пропуску в тій же ознаці, з рядком, що має пропуск:

$$L_{iy} = \frac{\#nonmissing\ in\ i,y}{distance(i,y)},$$

де в чисельнику знаходиться кількість непропущених пар значень в рядках і та у, в знаменнику - відстань між рядками. Обчислюються компетентності

всіх стовпців, які не мають пропуску в тому самому об'єкті, зі стовпцем, що має пропуск:

$$L_{iy} = |\ cor\ (i,y)\ |\ distance\ (i,y),$$

де $cor\ (i,y)$ - кореляція стовпців, $distance\ (i,y)$ - відстань між рядками. Серед всіх рядків і стовпців вибирається задану кількість рядків і стовпців з найбільшою компетентністю.

Далі за відомими значеннями в рядку і стовпці з замінним пропущеним значенням підбирається показник α ступеня обліку компетентності у зваженій сумі окремо для рядків і стовпців, мінімізуючи:

$$\sum_i |a_{ik} - b_{ik}| \to min,$$

передбачення b_{ik} обраховується за формулами:

$$b_{ik} = \frac{\sum_{j=1}^{c-1} bl_{jk} L_{ij}^{\alpha}}{\sum_{j=1}^{c-1} L_{ij}^{\alpha}}$$

де bl_{jk} - прогноз для значень рядка (стовпчика) k за допомогою j рядка (стовпчика) лінійної регресії виду y = ax + b. Передбачення по рядках і стовпцях усереднюється:

$$b_{ik} = \frac{b_{ik}^{row} + b_{ik}^{col}}{2}.$$

Описана послідовність дій застосовується для кожного пропущеного значення в матриці.

Для різних прикладних задач були реалізовані різні модифікації описаного вище ZET-алгоритму, відмінними за своїм призначенням та режимами роботи. Таким чином алгоритм може працювати в таких режимах:

Заповнення усіх пропусків.

Заповнення тільки тих пропусків, очікувана помилка яких не перевищує певну величину.

Заповнення пропусків тільки на базі інформації, наявної у вихідній таблиці.

Заповнення кожного наступного пропуску з використанням вихідної інформації і прогнозованих значень раніше заповнених пропусків.

2.4 Висновок до розділу

У роботі були розглянуті найбільш поширені методи відновлення пропусків у даних, проведено їх порівняння між собою. Експерименти показали, що немає методу, який би переважав за якістю всі інші і міг би бути універсальним підходом в будь-якій задачі.

Вибір методу заповнення пропусків може залежати від типів ознак, в яких існують пропуски, від кількості об'єктів, що мають пропущені значення, і від

причини їх виникнення. У кожному завданні необхідний індивідуальний підбір методу обробки пропущених значень.

Прості методи заповнення пропусків (заповнення модою, середньою та спеціальною значенням) показують якість порівнянну з просунутими методами, що пророкують пропущене значення, тому застосування витратних за часом роботи методів може бути необґрунтованим в рішенні задач аналізу даних.

РОЗДІЛ 3 ДОСЛІДЖЕНЯ ТА АНАЛІЗ РЕЗУЛЬТАТІВ ОТРИМАНИХ У ХОДІ ПРОВЕДЕННЯ ЕКСПЕРИМЕНТІВ

3.1 Вступ до розділу

В цьому розділі буде розроблено програмний продукт з реалізацією вище описаних алгоритмів. Буде проведено ряд експериментів із заповнення пропусків даних з використанням даних алгоритмів. Вибірки були взяті як із вже існуючими пропусками так і з вибірками в яких пропуски створюватимуться штучно.

3.2 Розробка програмного продукту

З цілю реалізації описаних у другому розділі основних методів заповнення даних було реалізовано програмний продукт. Окрім алгоритмів відновлення пропусків, є необхідність розробки алгоритмів формування пропусків в даних для проведення більш детального аналізу якості алгоритмів на вибірках, які не містять натуральних пропусків.

Механізм формування пропусків штучно імітує утворення MCAR пропусків.

3.2 Проведення експериментів

В цьому пункті описано експерименти на наборах даних взятих з відкритих джерел.

3.2.1 Експерименти з штучно створеними пропусками

Для порівняння якості відновлення пропусків у даних використовувалися набори даних, взятих з UCI Machine Learning Repository [Lichman].

Використання даних без пропусків мотивовано тим, що в такому випадку можна управляти кількістю пропущених значень. Це дозволяє порівняти методи для різної частоти пропусків серед наявних значень. Крім того, інформація про справжні значеннях пропусків дозволяє порівняти відновлені значення безпосередньо, а не тільки через якість роботи алгоритму машинного навчання на отриманих даних.

При використанні даних без пропусків необхідно створити пропущені значення штучно. Для цього існують різні стратегії. У статті [Gupta, Lam] пропонується створювати пропуски для випадкових обраних об'єктів в декількох ознаках, що мають максимальну зв'язок з цільовою змінною по хі-квадрат критерію. В інших статтях [Towards missing data imputation], [Wohlrab, Furnkranz], [Zhou, Lim] видаляють значення випадково в деякому підмножині ознак. Існують і більш складні стратегії створення пропущених значення, наприклад, пропущене значення в одному ознаці, якщо в іншому значення вище деякого порога. Однак неможливо підібрати метод, що імітує всі можливі сценарії пропущених значень. З цієї причини і через простоти в даній роботі використовувалося створення пропущених значень випадковим чином (MCAR).

Як наборів даних, що не мають натуральні пропуски, використовувалися:

Segment (Image Segmention) – необхідно класифікувати зображення за ознаками високого рівня. 2310 об’єктів і 19 ознак, всі є кількісними.

Breast Tissue Data Set – набір даних аналізів імпедансу. Імпеданс – електричний опір ділянок шкірного покрову людини, використовується для виявлення типу шкірного покрову та для діагностики різного роду захворювань. Саме для класифікації захворювань був створений цей набір даних. До нього входять 10 показників для 106 об’єктів, усі показники є кількісними.

Використовувалися наступні параметри кожного з методів відновлення (при можливості їх завдання):

1. Заміна спеціальним значенням: для застосування випадкового лісу пропуск замінювався -1, для логістичної регресії і методу найближчого сусіда - 0.

2. сингулярного розкладання: ранг апроксимуючої матриці в два рази менше кількості ознак, максимальне число ітерацій одно 10.

3. Метод k найближчих сусідів: k = 5, метрика простору L2.

4. Випадковий ліс: 10 дерев, максимальне число ітерацій - 3.

5. Лінійна регресія: максимальне число ітерацій - 3.

6. EM-алгоритм: 1 суміш нормального розподілу з повною матрицею коваріації.

7. Метод k середніх: 8 кластерів, максимальне число ітерацій - 3.

8. ZET: число компетентних строк - 6, число компетентних стовпців - 4.

Використання описаних значень параметрів методів пояснюється тим, що експерименти з їх варіюванням показали, що якість кінцевої класифікації слабо залежить від конкретних значень параметрів. Тому значення ці установки не були підбиралися для кожної з задач.

Для всіх методів, за винятком заміною середнім значенням і модою, вироблялося округлення до найближчого значення в масиві.

Для оцінки близькості відновленого набору даних до істинного (у випадку з створенням штучних пропусків) використовувалося середньоквадратичне відхилення.

Залежність середньоквадратичної похибки від частки пропущених даних у вибірці Segment зображена на рисунку 2.1.

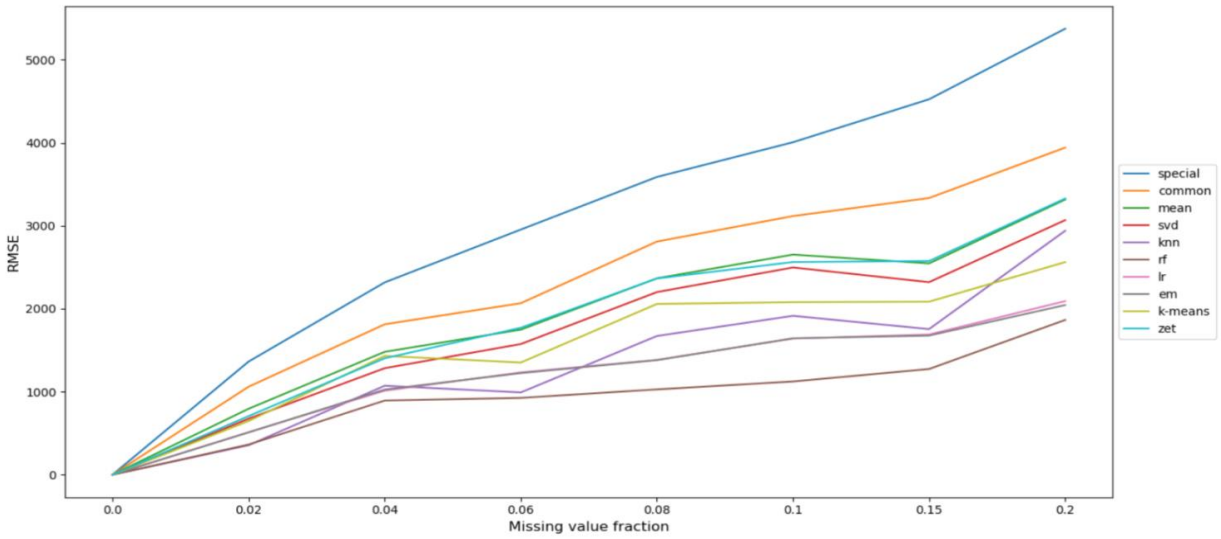


Рисунок 2.1 - Середньоквадратична похибка на вибірці Segment

Для вимірювання якості відновлення пропусків створювалися набори даних з частками пропусків від 0% до 20%.

З порівняння був виключений метод, який ігнорує об'єкти з пропущеними значеннями, через можливу роботу тільки з малою часткою пропусків в об'єктах.

Для даної вибірки кращий результат показав алгоритм відновлення пропусків за допомогою випадкового лісу. Хороший результат показують методи, засновані на k найближчих сусідів, EM-алгоритмі і лінійної регресії. Низький результат має алгоритм заміни середнім.

Заміна спеціальним значенням показала найгірший результат при застосуванні логістичної регресії і методу k найближчих сусідів.

Наступний експеримент проводився на вибірці Breast Tissue Data Set, результати якого зображені на рисунку 2.2.

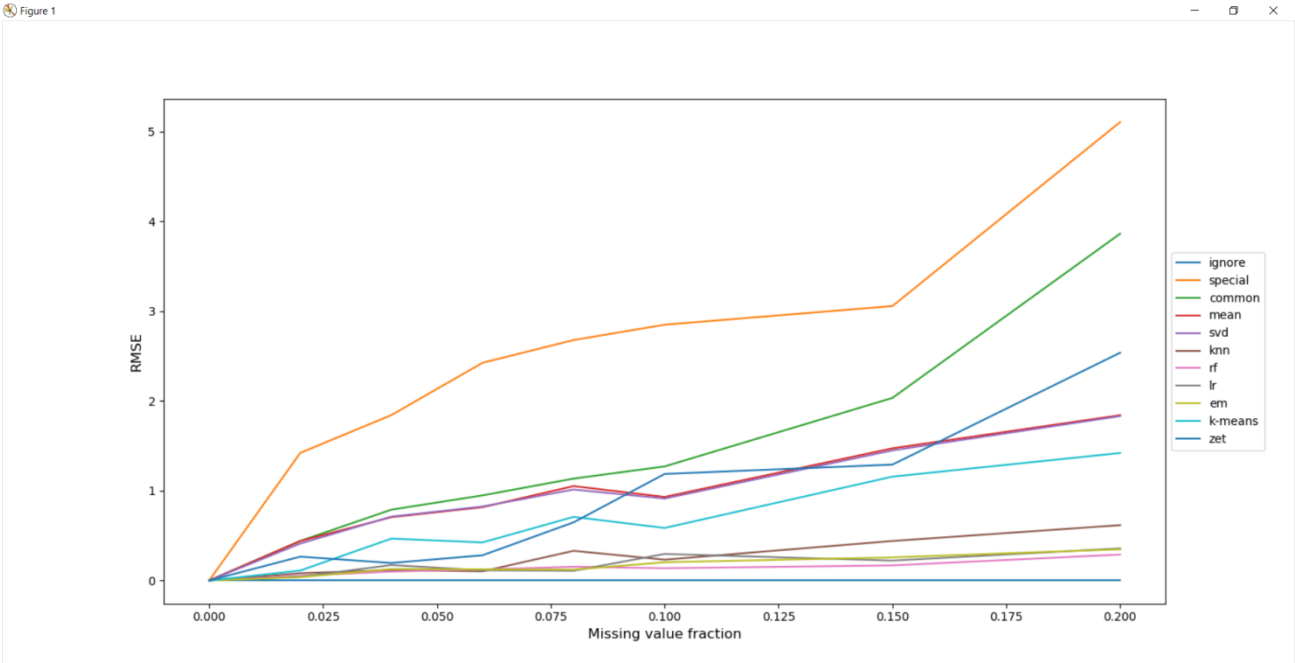


Рисунок 2.2 - Середньоквадратична похибка на вибірці Breast Tissue Data Set

Для цієї вибірки двома кращими та близькими за значенням середньоквадратичного відхилення виявилися алгоритми випадкового лісу та лінійної регресії, також гарні результати показали EM-алгоритм та метод k ближчих сусідів. Значно більші значення середньоквадратичного відхилення показали метод k середніх, SVD-алгоритм, метод заповнення середнім та Zet-алгоритм. Найгіршими виявилися методи заповнення модою та спеціальним значенням.

На прикладі цієї вибірки можна побачити що прості в реалізації методи суть яких полягає в заповнені пропусків мірами центральної тенденції або спеціальними значеннями показують значно гірші результати аніж більш складні методи.

Цікаву властивість можна помітити у методу заповнення середнім та SVD-алгоритму. Вони майже не відрізняються за значеннями середньоквадратичної похибки.

3.2.2 Експерименти з натуральними пропусками

Також було проведено експерименти над вибіркою, яка містила натуральні пропуски даних. Як приклад такої вибірки було взято набір даних Air Quality Data Set, який містить з показники газового пристрою, розміщеного на полі в Італійському місті. Середньочасові показники записані разом з показниками концентрації газу отриманими від сертифікованого пристрою. Набір даних містить 9358 екземплярів погодинних усереднених відповідей з п’яти хімічних датчиків оксиду металу, вбудованих у пристрій для хімічного сенсора якості повітря. Прилад розміщувався на полі у значно забрудненій місцевості, на рівні дороги, в межах міста. Дані записувались з березня 2004 року по лютий 2005 року (один рік), що представляють найдовші вільно доступні записи про реакції хімічних датчиків якості повітря на місцях. Концентрація CO, неметалічних вуглеводнів, бензолу, загального оксиду азоту (NOx) та діоксиду азоту (NO2) була забезпечена поруч розташованим сертифікованим аналізатором. Відсутні значення позначені тегом значення -200. Оскільки початковий набір даних містив цілих 9358 екземплярів, що значно збільшувало час роботи програми, було вирішено зменшити кількість екземплярів до 3000. Відсоток натуральних пропущених даних – 8.5

Для порівняння методів заповнення даних на вибірці з натуральними пропусками було побудована таблиця з математичними сподіваннями (табл.3.1) та дисперсіями кожного показника (табл.3.2) та відповідних методів.

Таблиця 3.1 – Математичні сподівання

	ignore	**special**	**common**	**mean**	**svd**
CO(GT)	2,162125	1,7297	2,0497	**2,162125**	2,1174
PT08.S1(CO)	1145,712	1122,225	1143,801	**1145,712**	**1145,764**
NMHC(GT)	218,8118	99,997	135,835	**218,8118**	**218,874**
C6H6(GT)	10,23014	10,02042	10,10336	**10,23014**	**10,23303**
PT08.S2(NMHC)	947,1089	927,6931	942,3148	**947,1089**	**947,1348**
NOx(GT)	134,5636	107,449	115,7105	**134,5636**	**132,1225**
PT08.S3(NOx)	963,8149	944,0567	959,0781	**963,8149**	**963,8025**
NO2(GT)	96,86788	77,349	96,8945	**96,86788**	**95,5415**
PT08.S4(NO2)	1600,405	1567,597	1596,153	**1600,405**	**1600,378**
PT08.S5(O3)	982,497	**962,3558**	972,3752	**982,497**	**982,5315**
T	17,51508	17,15603	**17,44661**	**17,51508**	**17,51418**

RH	48,10241	47,11631	47,65956	**48,10241**	48,0944
AH	0,908318	0,889697	0,897394	**0,908318**	**0,908168**

Кінець таблиці 3.1

	knn	**rf**	**lr**	**em**	**k-means**	**zet**
CO(GT)	2,09955	2,08065	**2,08325**	**2,08365**	2,09805	1,7661
PT08.S1(CO)	1146,14	**1145,758**	1146,439	1146,66	1145,262	1137,308
NMHC(GT)	221,35	217,579	226,188	219,3265	209,036	118,3845
C6H6(GT)	10,2128	10,20904	**10,23241**	10,25396	10,20619	10,02994
PT08.S2(NMHC)	946,6791	**947,105**	**947,348**	948,1786	946,9324	936,6516
NOx(GT)	130,4725	**131,1715**	130,6725	130,649	**131,2995**	108,9335
PT08.S3(NOx)	964,9787	961,9372	**963,5098**	962,7898	962,8911	953,3125
NO2(GT)	94,746	95,8735	94,68	94,7225	**95,626**	79,7555
PT08.S4(NO2)	1598,812	1600,237	**1600,693**	1601,391	**1599,833**	1587,174
PT08.S5(O3)	981,6596	983,3151	983,7048	984,5095	981,672	967,7473
T	**17,50746**	17,49706	**17,50466**	**17,51219**	**17,5281**	17,28108
RH	47,99715	48,12613	48,12957	**48,10872**	**48,0834**	47,3044
AH	0,905682	**0,908701**	**0,908105**	**0,908174**	**0,908501**	0,897394

Таблиця 3.2 - Дисперсії

	ignore	**special**	**common**	**mean**	**svd**
CO(GT)	1,329	1,470	1,209	1,188	1,235
PT08.S1(CO)	228,551	278,452	226,581	226,196	**226,197**
NMHC(GT)	204,460	176,012	157,768	138,177	138,226
C6H6(GT)	7,112	7,187	**7,093**	7,039	7,039
PT08.S2(NMHC)	259,208	289,537	**258,669**	256,536	256,537
NOx(GT)	77,355	87,706	**78,656**	69,120	71,896
PT08.S3(NOx)	254,688	286,701	**254,181**	252,062	252,062
NO2(GT)	32,106	48,307	28,688	28,688	30,295
PT08.S4(NO2)	279,923	358,058	278,593	277,037	277,037
PT08.S5(O3)	364,202	386,414	367,179	360,448	360,448
T	5,402	5,895	5,367	5,346	**5,346**
RH	16,345	17,555	**16,464**	16,176	16,177
AH	0,201	0,237	0,213	0,199	0,199

Кінець таблиці 3.2

	knn	**rf**	**lr**	**em**	**k-means**	**zet**
CO(GT)	**1,305**	**1,313**	**1,317**	**1,325**	1,293	1,456
PT08.S1(CO)	**227,814**	**226,586**	**226,572**	**226,951**	**226,217**	233,540
NMHC(GT)	**189,591**	**193,547**	**189,436**	176,689	166,600	176,864
C6H6(GT)	7,072	7,040	7,041	**7,062**	7,041	**7,174**
PT08.S2(NMHC)	**257,962**	256,536	256,599	257,416	256,539	266,531
NOx(GT)	75,483	74,685	75,877	**76,227**	74,990	86,117
PT08.S3(NOx)	**254,512**	252,492	252,134	252,763	252,143	262,313
NO2(GT)	31,478	31,046	**31,962**	**32,015**	30,967	45,021
PT08.S4(NO2)	**279,626**	277,079	277,149	**277,916**	277,066	291,750
PT08.S5(O3)	**363,668**	**361,298**	360,804	**361,853**	360,493	374,597
T	5,363	5,348	**5,350**	5,349	5,347	5,586
RH	**16,316**	16,177	16,183	16,186	16,177	17,092
AH	**0,202**	**0,199**	**0,199**	**0,199**	**0,199**	0,213

Оскільки не існує методу за яким можна визначити якість методів заповнення даних на вибірці з натуральними пропусками було вирішено визначити та порівняти середнє значення та дисперсію для кожного методу. Як видно з наведених вище таблиць, де жирним шрифтом виділенні значення середнього та дисперсії, які ближчі до початкових та/або близькі один до одного, близькими за значеннями виявилися алгоритми n найближчих сусідів,

алгоритм випадкового лісу, EM-алгоритм. Близькі за значенням середнього, але не близьким за дисперсією виявився SVD алгоритм.

3.3 Аналіз отриманих результатів

Кращий результат показав алгоритм відновлення пропусків за допомогою випадкового лісу. Гарні результати показують методи, засновані на k найближчих сусідів, EM-алгоритмі і лінійної регресії.

Прості алгоритми в основі яких лежить механізм заміни пропусків на певні значення такі як мода, медіана та спеціальні значення дають переважно більшу похибку.

Якщо порівнювати локальні та глобальні алгоритми за їх ефективністю то певної переваги однієї з груп не було виявлено.

РОЗДІЛ 4 ФУНКЦІОНАЛЬНО-ВАРТІСНИЙ АНАЛІЗ ПРОГРАМНОГО ПРОДУКТУ

4.1 Постановка задачі проектування

Спроектувати програмний продукт для аналізу методів розв’язання задачі заповнення пропусків даних.

4.2 Обґрунтування функцій та параметрів програмного продукту

Виходячи з конкретних цілей, які реалізуються :

F1 – завантаження даних - а) завантаження даних, б) штучне створення даних.

F2 – обробка даних - а) видалення зайвих атрибутів, б) заміна якісними значеннями.

F3 – збереження результатів роботи - а) запис результатів у сховище та вивід користувачу, б) вивід користувачу.

Виходячи з представлених варіантів будуємо морфологічну карту (рис.4.1).

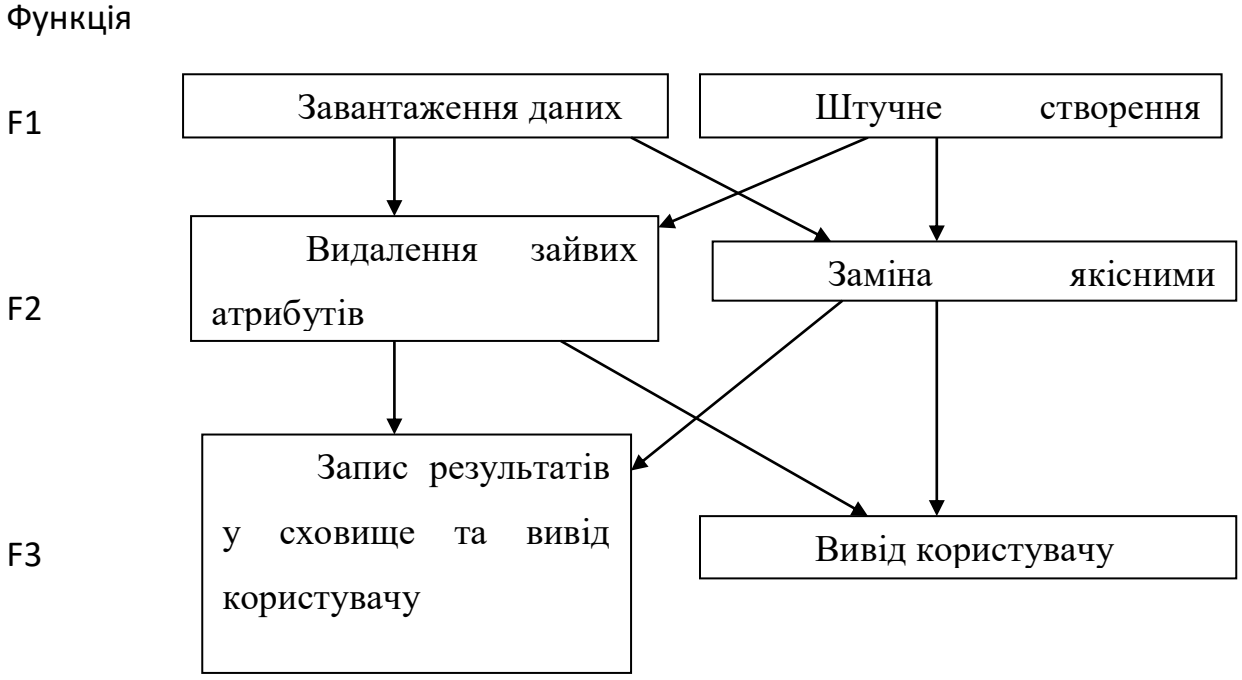


Рисунок 4.1 - Морфологічна карта.

Спираючись на карту була побудована позитивно-негативна матриця (табл. 4.1)

Таблиця 4.1 - Позитивно-негативна матриця

Основна функція	Варіант реалізації	Переваги	Недоліки
F1	А	Практичне застосування	Необхідність додаткової обробки даних
	Б	Зручність	Відсутність практичного застосування
F2	А	Легкість реалізації	Можливі значні втрати даних
	Б	Легкість реалізації	Неможливість застосування деяких алгоритмів
F3	А	Можливість спостереження за результатами попередніх експериментів	Потрібне надійне сховище даних, складність реалізації
	Б	Легкість реалізації	Недоступність результатів попередніх тестувань

Для характеристики прототипу програмного додатку використовуємо параметри X1-X4. На основі даних, що представлені у літературі, визначаємо мінімальні, середні отримуванні та максимально допустимі значення (табл. 4.2). Функція F1 впливає на параметр X1, функція F2 на X2, функція F3 впливає одночасно на два параметри X3 та X4.

Таблиця 4.2 - Система параметрів додатку

Найменування параметру	Позначення параметру	Значення параметру		
		Мінімальне	Середнє	Максимальне
Час розробки, людина*год	X1	352	528	704
Час роботи алгоритму, мс	X2	100	550	1000
Рекомендована швидкість запису на диск, МБ/с	X3	0	32	64
Об’єм пам’яті для збереження даних, МБ	X4	0	16	32

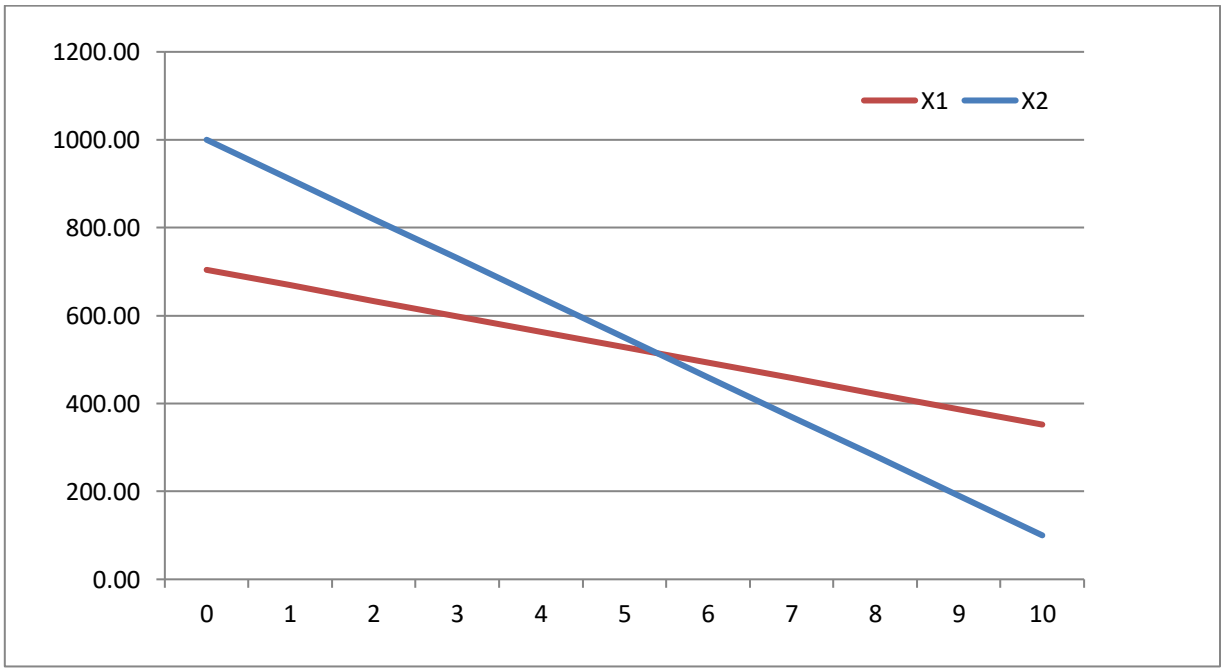


Рисунок. 4.2 - Значення параметрів X1, X2

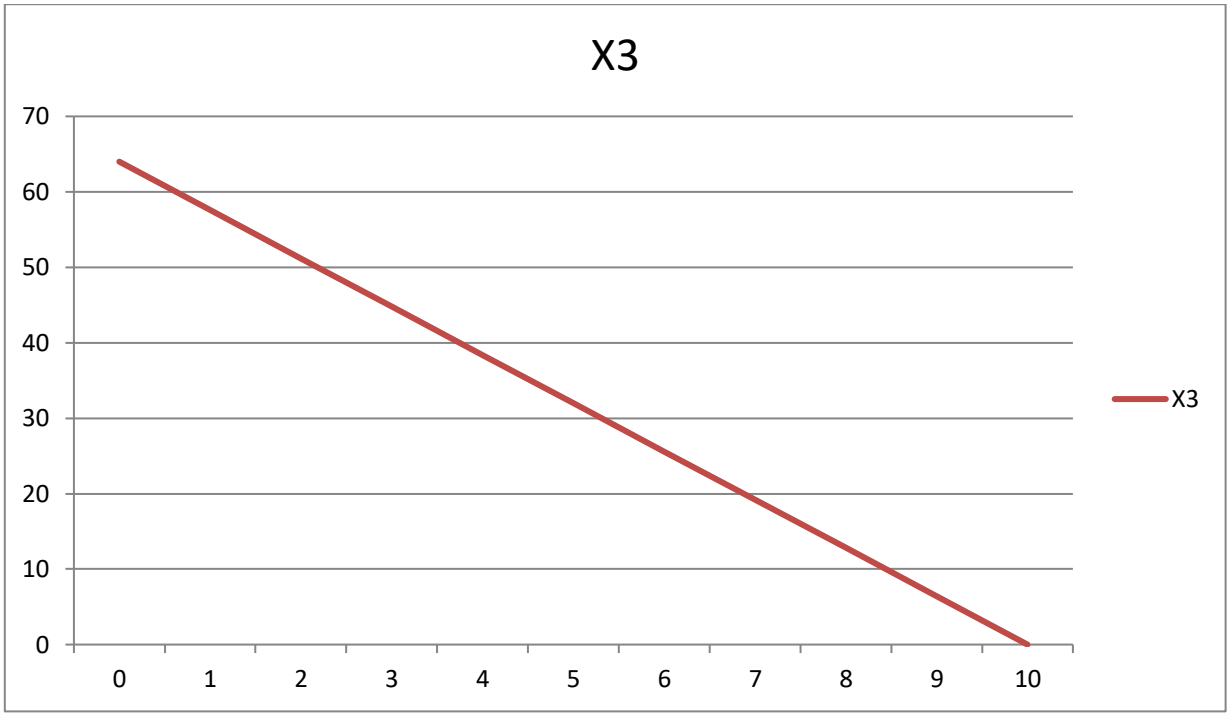


Рисунок. 4.3 - Значення параметра X3

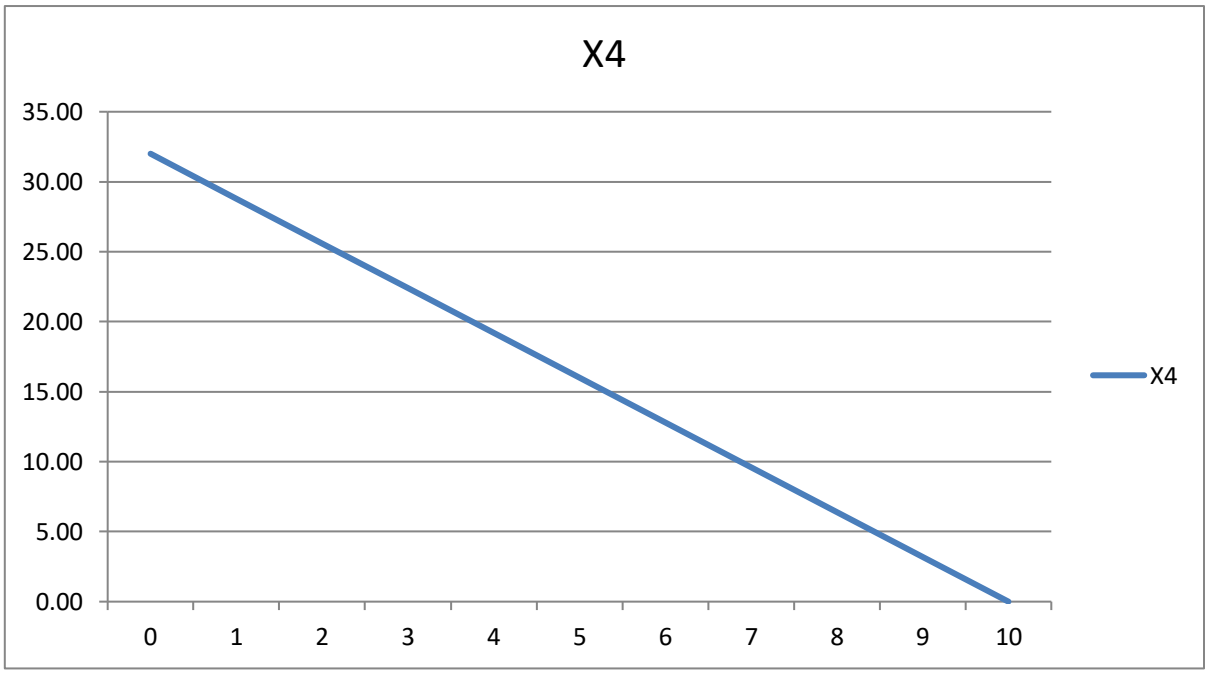


Рисунок.4.4 - Значення параметрів X4

Вагомість параметрів оцінюється за допомогою методів попарного зрівняння. Ранги варіюються від 1 до 4.Результати наведені в табл. 4.3-4.4

Таблиця. 4.3 - Результат оцінки параметрів

Параметр	Ранг параметру по оцінці експерта							Сума рангів, R_i	ВідхиленняΔ_i	Квадрат відхилення, $(\Delta_i)^2$
	1	2	3	4	5	6	7			
X1	3	2	1	1	2	1	2	12	-5.5	30.25
X2	1	1	2	2	3	2	1	12	-5.5	30.25
X3	4	4	4	4	4	4	4	28	10.5	110.25
X4	2	3	3	3	1	3	3	18	0.5	0.25
Разом	10	10	10	10	10	10	10	70	0	171

За 1 приймається найбільший ранг, за 4 – найменший.

Таблиця 4.4 - Попарне зрівняння параметрів

Параметри	Експерти							Кінцева оцінка	Числове значення
	1	2	3	4	5	6	7		
X1 та X2	<	<	>	>	>	>	<	>	1.5
X1 та X3	>	>	>	>	>	>	>	>	1.5
X1 та X4	<	>	>	>	<	>	>	>	1.5
X2 та X3	>	>	>	>	>	>	>	>	1.5
X2 та X4	>	>	>	>	<	>	>	>	1.5
X3 та X4	<	<	<	<	<	<	<	<	0.5

Для перевірки степені достовірності експертних оцінок, визначимо наступні параметри:

а) сума рангів кожного з параметрів і загальна сума рангів:

$$R_i = \sum_{j=1}^{N} r_{ij} R_{ij} = \frac{Nn(n+1)}{2} = 70$$

де N – число експертів,

n – кількість параметрів;

б) середня сума рангів:

$$T = \frac{1}{n} R_{ij} = 17{,}5$$

в) відхилення суми рангів кожного параметра від середньої суми рангів:

$$\Delta_i = R_i - T$$

Сума відхилень по всім параметрам повинна дорівнювати 0;

Загальна сума квадратів відхилення:

$$S = \sum_{i=1}^{N} \Delta_i^2 = 171$$

Визначимо коефіцієнт конкординації: $W = \frac{12S}{N^2(n^3-n)} = \frac{12*171}{7^2(4^3-4)} = 0.69 > W_k = 0.67$

Так як коефіцієнт конкординації більше нормативного, результати вважають достовірними.

Числове значення, що визначає ступінь переваги *i*–го параметра над *j*–тим, a_{ij} визначається по формулі:

$$a_{ij} = \begin{cases} 1.5 \text{ при } X_i > X_j \\ 1.0 \text{ при } X_i = X_j \\ 0.5 \text{ при } X_i < X_j \end{cases}$$

З отриманих числових оцінок переваги складемо матрицю А=$\| a_{ij} \|$.

Для кожного параметра зробимо розрахунок вагомості $K_{вi}$ за наступними формулами:

$$K_{\text{вi}} = \frac{b_i}{\sum_{i=1}^{n} b_i},$$

де $b_i = \sum_{j=1}^{n} a_{ij}$.

Відносні оцінки розраховуються декілька разів доти, поки наступні значення не будуть незначно відрізнятися від попередніх (менше 2%). На другому і наступних кроках відносні оцінки розраховуються за наступними формулами:

$$K_{\text{ві}} = \frac{b_i'}{\sum_{i=1}^{n} b_i'}$$

де $b_i' = \sum_{j=1}^{n} a_{ij} b_j$.

Розрахунок вагомості параметрів наведено в табл. 4.5

Таблиця 4.5 - Розрахунок вагомості параметрів

Параметри	Параметри xj				Перший крок		Другий крок		Третій крок	
	X1	X2	X3	X4	b_i	$К_{ві}$	b_i	$К_{ві}$	b_i	$К_{ві}$
X1	1	1,5	1,5	1,5	5,5	0,34	21,25	0,33	85,375	0,34
X2	0,5	1	1,5	1,5	4,5	0,28	16,25	0,26	66,625	0,27
X3	0,5	0,5	1	0,5	2,5	0,16	9,25	0,14	46,625	0,18
X4	0,5	0,5	1,5	1	3,5	0,22	17,25	0.27	49,875	0,21
Загалом:					16	1	64	1	248,5	1

Враховуючи дані з порівнянь варіантів реалізацій функцій можна виключити з реалізацій функцій наступний варіант: F2(б).

Залишаються наступні варіанти:

F1(а)=>F2(а)=>F3(а)

F1(а)=>F2(а)=>F3(б)

F1(б)=>F2(а)=>F3(а)

F1(б)=>F2(а)=>F3(б)

Аналіз рівня якості варіантів реалізації функцій (табл. 4.6).

Таблиця 4.6 - Розрахунок показників рівня якості варіантів реалізації основних функцій ПП.

Основна функція	Варіант реалізації	Параметри	Абсолютне значення параметру	Бальна оцінка параметру	Коефіцієнт вагомості параметру	Коефіцієнт якості
F1	а)	X1	600	2,95	0,34	1,003
	б)		500	5,79	0,34	1,9686
F2	а)	X2	150	9,44	0,27	2.5488
F3	а)	X3	32	5	0,18	0,9
		X4	20	3,75	0,21	0,7875
	б)	X3	32	5	0,18	0,9
		X4	4	8,75	0,21	1,8375

Обрахуємо коефіцієнти якості кожного з варіантів розробки: -

$K_{я1} = 1{,}003 + 2{,}5488 + 0{,}9 + 0{,}7875 = 5{,}2393$

$K_{я2} = 1{,}003 + 2{,}5488 + 0{,}9 + 1{,}8375 = 6{,}2893$

$K_{я3} = 1{,}9686 + 2{,}5488 + 0{,}9 + 0{,}7875 = 6{,}2049$

$K_{я4} = 1{,}9686 + 2{,}5488 + 0{,}9 + 1{,}8375 = 7{,}2549$

Оскільки варіант 4 має найбільший коефіцієнт якості, він є найкращим.

4.3 Економічний аналіз варіантів розробки

Для оцінки трудомісткості розробки спочатку проведемо розрахунок трудомісткості. Усі варіанти мають наступні основні завдання:

1) Видалення зайвих атрибутів.

2) Вивід результатів на екран.

Також кожний з варіантів має додаткові завдання, які є реалізаціями розгалужених варіантів розробки незалежного модуля. Далі наведено варіанти додаткових завдань(два завдання, які мають номери 1 в реалізаціях)

3.1) Завантаження даних з пам’яті.

3.2) Штучне створення даних.

4.1) Збереження отриманих результатів.

В варіанті 1 присутні наступні додаткові завдання під номерами 3.1 та 4.1

В варіанті 2 присутні наступні додаткові завдання під номерами 3.1

В варіанті 3 присутні наступні додаткові завдання під номерами 3.2 та 4.1

В варіанті 4 присутні наступні додаткові завдання під номерами 3.2

За ступенем новизни до групи Б відноситься завдання 1, 3.2, до групи В відносяться завдання 2, до групи Г завдання 4.1, 3.1

За складністю алгоритмів до групи 2 відносяться завдання 1, 3.2 до групи 3 відноситься завдання 2, 3.1, 4.1

Спираючись на норми розрахункового часу визначимо трудомісткість. Вона складає для першого завдання $T_р$=43 людино-днів. Поправочний коефіцієнт, що враховує новизну ПП складає $К_н$=1 (нормативно-довідкова інформація). Оскільки під час виконання даного завдання використовуються новостворенні модулі, врахуємо це за допомогою коефіцієнта $К_{ст}$ = 0,6. Коефіцієнти $К_м$ і $К_{ст.м}$, які враховують відповідно програмування на мові низького рівня та розробку стандартного програмного забезпечення, для всіх завдань дорівнюють 1.

Загальна трудомісткість обчислюється як

$$Т_О = Т_Р * К_Н * К_М * К_{СТ} * К_{СТ.М},$$

де $Т_Р$ – трудомісткість розробки ПП;

$К_Н$ – поправочний коефіцієнт;

$К_М$ – коефіцієнт рівня мови програмування;

$К_{СТ}$ – коефіцієнт використання стандартних модулів і прикладних програм;

$К_{СТ.М}$ – коефіцієнт стандартного математичного забезпечення

Повна трудомісткість першого завдання(складність – 2, новизна – Б):

$Т_1 = 33 * 1 * 0,8 = 26,4$

Аналогічно для завдання 2(складність – 3, новизна – В): –

$Т_р$= 21; $К_Н$ = 0,7; $К_{ст}$ = 0,6; $Т_2 = 21 * 0,7 * 0,6 = 8,82$

Аналогічно для завдання 3.1(складність – 3, новизна – Г):

$Т_р$= 23; $К_Н$ = 1; $К_{ст}$ = 0,6; $Т_3 = 23 * 1 * 0,6 = 13,8$

Аналогічно для завдання 3.2(складність – 2, новизна – Б)

$Т_р$= 38; $К_Н$ = 0,7; $К_{ст}$ = 0,8; $Т_4 = 38 * 0,7 * 0,8 = 21,28$

Аналогічно для завдання 4.1(складність – 3, новизна – Г):

$Т_р$= 21; $К_Н$ = 1; $К_{ст}$ = 0,6; $Т_5 = 21 * 1 * 0,6 = 12,6$

Визначимо повну трудомісткість варіантів(людино-днів):

$$Т_1 = 26,4 + 8,82 + 13,8 + 12,6 = 61,62$$

$$Т_2 = 26,4 + 8,82 + 13,8 = 49,02$$

$$Т_3 = 26,4 + 8,82 + 21,28 + 12,6 = 69,1$$

$$Т_4 = 26,4 + 8,82 + 21,28 = 56,5$$

Найбільш трудомісткими завданнями є 1 та 3, найбільш трудомісткий варіант - 3

Далі вважається, що робочий день складає 8 годин, в тижні п’ять робочих днів. В розробці бере участь один програміст з окладом 13500 грн та інженер даних з окладом 10000 грн.

Визначимо зарплату за годину за формулою:

$$С_ч = \frac{М}{T_m \cdot t} грн$$

де М – місячний оклад працівників;

T_m– кількість робочих днів в місяць;

t – кількість робочих годин в день.

Середня заробітна плата за годину:

$$С_ч = \frac{13500 + 10000}{2 * 22 * 8} = 66,76$$

Тоді заробітна плата для кожного з варіантів реалізації(грн):

$$С_{ЗП} = 66,76 * 8 * 61,62 = 32910$$

$$С_{ЗП} = 66,76 * 8 * 49,02 = 26180$$

$$С_{ЗП} = 66,76 * 8 * 69,1 = 36904,93$$

$$С_{ЗП} = 66,76 * 8 * 56,5 = 30175,52$$

Відрахування на соціальне страхування(22%)(грн):

$$С_{ВІД} = 32910 * 0,22 = 7240,2$$

$$С_{ВІД} = 26180 * 0,22 = 5759,6$$

$$С_{ВІД} = 36904,93 * 0,22 = 8119,08$$

$$С_{ВІД} = 30175,52 * 0,22 = 6638,61$$

Далі розрахуємо витрати на оплату однієї машино-години. Враховуючи, що вона обслуговує одного спеціаліста з окладом 13500 грн та одного з окладом 10000 грн з коефіцієнтом зайнятості 0,6, то для двох машин отримаємо

$С_г = 12 * 13500 * 0,6 + 12 * 10000 * 0,6 = 169200$ грн

Враховуючи додаткову заробітну плату

$$С_{ЗП} = 169200 * (1 + 0,4) = 270720$$

Відрахування на соціальне страхування 22%

$$С_{ВІД} = 270720 * 0,22 = 59558,4$$

Розрахуємо амортизаційні підрахунки (амортизація 25%, вартість ЕОМ 25000 грн)

$С_А = К_{ТМ} * К_А * Ц_{ПР} = 1,15 * 0,25 * 25000 = 7187,5$ грн

Розрахуємо витрати на ремонт та профілактику:

$$С_Р = К_{ТМ} * Ц_{ПР} * К_Р = 1,15 * 25000 * 0,05 = 1437,5грн$$

Розрахуємо ефективний годинний фонд часу ПК за рік

$$Т_{ЕФ} = (365 - 142 - 16) * 8 * 0,8 = 1324,8 год$$

Розрахуємо витрати на електроенергію

$$С_{ЕЛ} = 1324{,}8 * 0{,}6 * 0{,}6 * 1{,}75 * 3 = 2191{,}33 \text{ грн}$$

Накладні витрати рівні:

$$С_{Н} = 25000 * 0{,}67 = 16750 \text{ грн.}$$

Отже експлуатаційні витрати(грн):

$$С_{ЕКС} = 270720 + 59558{,}4 + 7187{,}5 + 1437{,}5 + 2191{,}33 + 16750 = 3357844{,}73$$

Тоді собівартість однієї машино-години ЕОМ дорівнюватиме:

$$С_{М-Г} = \frac{3357844{,}73}{1324{,}8} = 270{,}11 \text{ грн/год}$$

Враховуючи, що всі роботи ведуться на ЕОМ, витрати на оплату машинного часу:

$$С_{м} = 270{,}11 * 8 * 61{,}62 = 133153{,}43$$

$$С_{м} = 270{,}11 * 8 * 49{,}02 = 105926{,}34$$

$$С_{м} = 270{,}11 * 8 * 69{,}1 = 149316{,}81$$

$$С_{м} = 270{,}11 * 8 * 56{,}5 = 122089{,}72$$

Накладні витрати відповідно

$$С_{Н} = 133153{,}43 * 0{,}67 = 89212{,}8$$

$$С_{Н} = 105926{,}34 * 0{,}67 = 70970{,}64$$

$$С_{Н} = 149316{,}81 * 0{,}67 = 100042{,}26$$

$$С_{Н} = 122089{,}72 * 0{,}67 = 81799{,}63$$

Розрахуємо повну вартість розробки за варіантами:

$$С_{ПП} = 32910 + 7240{,}2 + 133153{,}43 + 89212{,}8 = 262516{,}43$$

$$С_{ПП} = 26180 + 5759{,}6 + 105926{,}34 + 70970{,}64 = 208836{,}58$$

$$С_{ПП} = 36904{,}93 + 8119{,}08 + 149316{,}81 + 100042{,}26 = 294383{,}08$$

$$С_{ПП} = 30175{,}52 + 6638{,}61 + 122089{,}72 + 81799{,}63 = 240703{,}48$$

4.4 Вибір кращого варіанта ПП техніко-технічного рівня

$$K_{я1} = 1{,}003 + 2{,}5488 + 0{,}9 + 0{,}7875 = 5{,}2393$$

$$K_{я2} = 1{,}003 + 2{,}5488 + 0{,}9 + 1{,}8375 = 6{,}2893$$

$$K_{я3} = 1{,}9686 + 2{,}5488 + 0{,}9 + 0{,}7875 = 6{,}2049$$

$$K_{я4} = 1{,}9686 + 2{,}5488 + 0{,}9 + 1{,}8375 = 7{,}2549$$

Розрахуємо коефіцієнт техніко-економічного рівня

$$К_{ТЕР1} = \frac{5{,}2393}{262516{,}43} = 1{,}99 * 10^{-5}$$

$$К_{ТЕР2} = \frac{6{,}2893}{208836{,}58} = 3{,}012 * 10^{-5}$$

$$К_{ТЕР3} = \frac{6{,}2049}{294383{,}08} = 2{,}1 * 10^{-5}$$

$$К_{ТЕР4} = \frac{7{,}2549}{240703{,}48} = 3{,}014 * 10^{-5}$$

4.5 Висновки до розділу

Отже враховуючи всі дослідження, що описані вище, можна сказати, що 4 варіант реалізації є найбільш оптимальним зі сторони якісно-економічної оцінки. Його коефіцієнт техніко-економічного рівня складає $3{,}014 * 10^{-5}$.

Розробка цього варіанту передбачає такі обов'язкові завдання як:

- Видалення зайвих атрибутів.
- Вивід результатів на екран.

Серед завдань між якими ставився вибір в даному варіанті реалізовані такі завдання:

- Штучне створення даних.

ВИСНОВКИ

В даній роботі було описано існуючі методи заповнення даних та проведено ряд експериментів. Також було порівняна ефективність даних алгоритмів на вибірках з пропусками даних.

Кращий результат показав алгоритм відновлення пропусків за допомогою випадкового лісу. Гарні результати показують методи, засновані на k найближчих сусідів, EM-алгоритмі і лінійної регресії.

Прості алгоритми в основі яких лежить механізм заміни пропусків на певні значення такі як мода, медіана та спеціальні значення дають переважно більшу похибку.

Перспективою подальших досліджень є розробка більш універсальних та точних алгоритмів заповнення пропусків даних.

Список використаної літератури

1. Kalton G., Kasprzyk D. The Treatment of Missing Survey Data: Survey Methodology. 1968. V. 12. P. 1-16.

2. Двоенко С.Д. Неиерархический дивизимный алгоритм кластеризации. Автоматика и телемеханика. 1999. № 4. С. 117-124.

3. Загоруйко Н.Г., Йолкіна В.Н., Алгоритм заполнения пропусков в эмпирических таблицах (алгоритм Zet). Эмпирическое предсказание и распознавание образов. Новосибірськ, 1975. - Вид. 61. Вычислительные системы. С. 3-27.

4. Hastie T., Tibshirani R., Sherlock G., Eisen M., Brown P., Botstein D. Imuting Missing Data for Gene Expression Arrays. 1999. P. 1-3.

5. Dempster A. P., Laird N. M., Rubin D. B. Maximum likelihood from incomplete data via the EM algorithm. Journal of the Royal Statistical Society. 1977. V. 39. P. 1–38.

6. Воронцов К.В. Математические методы обучения по прецедентам (теория обучения машин). С. 32-37.

7. MacQueen J. Some methods for classification and analysis of multivariate observations. Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability. 1967. V. 1. P. 281–297.

8. Загоруйко. Н.Г. Прикладные методы анализа данных и знаний. Институт математики. 1999.

9. Lichman M. UCI machine learning repository. 2013. URL: http://archive.ics.uci. edu/ml.

10. Gupta A., Lam M. The weight decay backpropagation for generalizations with missing values. Annals of Operations Research. 1998. V. 78. P. 165–187.

11. D. Li, J. Deogun, W. Spaulding, B. Shuart Towards missing data imputation: A study of fuzzy k-means clustering method. Rough Sets and Current Trends in Computing. 2004. V. 3066. P. 573–579.

12. Wohlrab L., Furnkranz J. A comparison of strategies for handling missing values in rule learning. 2009.

13. Zhou X.-Y., Lim J. S. EM algorithm with GMM and naive bayesian to implement missing values. Advanced Science and Technology Letters. 2014. V. 46. P. 1–5.

ДОДАТОК А Ілюстративні матеріали до доповіді

Розв’язання задачі заповнення пропусків даних альтернативними методами

Керівник:
викладач кафедри ММСА
Кухарєв С.О.

Виконав:
студен 4 курсу, групи КА-64
Маркін І.Д.

Об’єкт дослідження:

- Задача заповнення пропусків даних

Предмет дослідження:

- Методи вирішення задачі заповнення пропусків даних

Мета дослідження:

- Проаналізувати існуючі алгоритми вирішення задачі заповнення пропусків даних, порівняти їх на конкретних вибірках.

Актуальність обраної теми:

У практичних завданнях аналізу даних вибірки часто містять в собі пропущені значення. Причини можуть бути різними, наприклад, відсутність відповіді респондента на конкретне запитання анкети, відмова датчика для вимірювань показника, помилки в програмному забезпеченні під час запису даних.

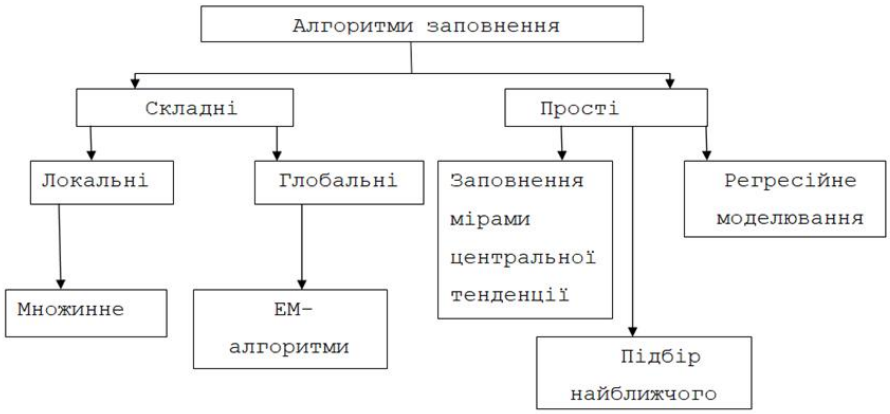
За рідкісним винятком алгоритми машинного навчання не працюють з вибірками, що мають пропущені значення. Тому виникає необхідність у процедурі заповнення даних.

У даній роботі описані основні методи відновлення пропусків даних, проведено їх порівняння на декількох наборах даних.

Види пропусків даних:

- MCAR (Missing Completely At Random) – механізм рівномірного формування пропусків, тобто ймовірність пропуску для кожного запису однакова.
- MAR (Missing At Random) – ймовірність пропуску може бути обрахована в залежності від іншої наявної в даних інформації.
- MNAR (Missing Not At Random) - механізм формування пропусків, при якому дані відсутні в залежності від невідомих чинників.

Класифікація методів:



Опис існуючих методів

- Ad-hoc методи – методи заповнення спеціальними значеннями такими як нулями, медіаною, середнім арифметичним, введення індикаторних змінних і тому подбні.

Опис існуючих методів

- Заміна за допомогою сингулярного розкладу:

Знаходиться сингулярний розкладання матриці:

$$X = U\Sigma V^*,$$

де $U^{n\times n}, V^{d\times d}$ – унітарні матриці, $\Sigma^{n\times d}$ – діагональна матриця з сингулярними числами.

Відновлення матриці:

$$X_{approx} = U\Sigma' V^*,$$

де Σ' отримана з Σ шляхом занулення всіх діагональних елементів крім k найбільших

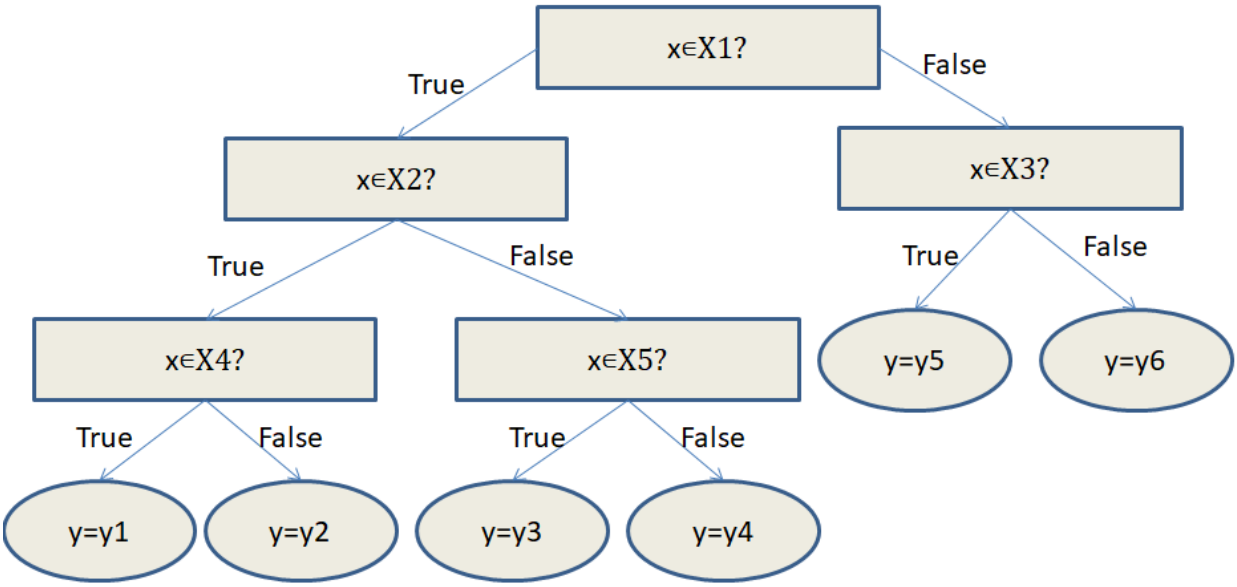
Опис існуючих методів

- Заміна за допомогою методу найближчих сусідів

З припущення про те, що близькі об'єкти за значеннями серед заповнених ознак близькі в ознаках, значення яких може бути пропущено (гіпотеза компактності), виникає застосування методу k найближчих сусідів для відновлення пропусків у даних. Реалізація аналогічна використанню класичного методу k найближчих сусідів за винятком того, передбачається відразу кілька пропущених ознак для кожного об'єкта.

Опис існуючих методів

- Заміна за допомогою випадкового лісу



Опис існуючих методів

- Заміна за допомогою ЕМ-алгоритму

$$p(x) = \sum_{k=1}^{K} \pi_k N(\mu_k, \Sigma_k), \pi_k \geq 0, \sum_{k=1}^{K} \pi_k = 1.$$

За середнім значенням і матрицями коваріації ознак обчислюються коефіцієнти регресії:

$$\beta_k = cov\,(X_{-j}, X_j)\,\Sigma_k^{-1},$$

де X_j- відновлювана ознака, X_{-j}- матриця без відновлюваної ознаки

За отриманими коефіцієнт обчислюються значення на місцях пропусків за іншими ознаками цього об'єкта: $X_j^k = X_{-j}\beta_k^T$

Підсумкове передбачення пропущеного значення усереднюється за розподілом:

$$X_j = \sum_{k=1}^{K} \pi_k X_j^k$$

Опис існуючих методів

- Заміна за допомогою методу k середніх

Аналогічно методу k найближчих сусідів передбачається, що близькі за одними ознаками об'єкти повинні бути близькі і за іншими ознаками. Однак на відміну від методу k найближчих сусідів шукають не найближчі сусіди для кожного об'єкта з пропущеними значеннями, а використовується інформація про центр кластера, в який потрапив конкретний об'єкт з пропусками.

Опис існуючих методів

- Алгоритм ZET

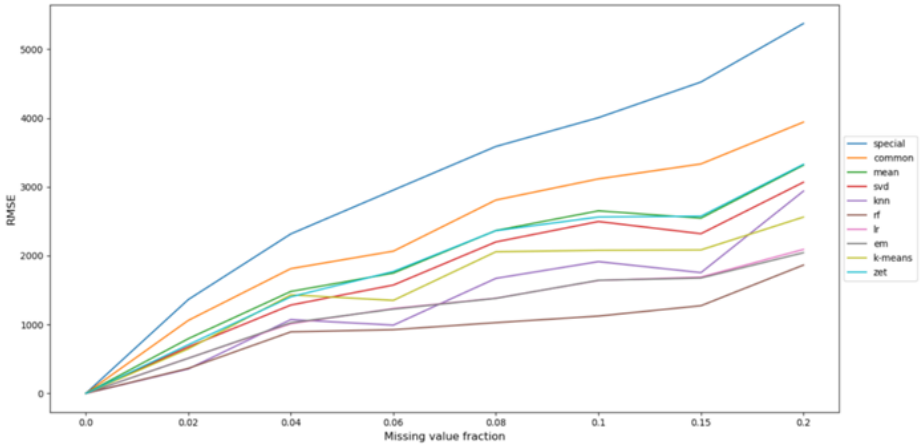
Основні етапи ZET алгоритму:

1. Попередня обробка початкових даних.
2. Прогнозування пропуску - виконується L раз:
2.1. Формування компетентної матриці.
2.2. Підбір параметрів моделі прогнозування.
2.3. Прогнозування пропуску.

Дані для експериментів

- Segment (Image Segmention) – база даних для класифікації зображень, які були сегментовані, для створення класифікації кожного пікселя. Пропуски створюються вручну.
- Air Quality Data Set - містить з показники газового пристрою, розміщеного на полі в Італійському місті. Набір даних містить натуральні пропуски.

Результати експериментів з штучно створеними пропусками



Результати експериментів з натуральними пропусками

	ignore	special	common	mean	svd	knn	rf	lr	em	k-means	zet
CO(GT)	2,162125	1,7297	2,0497	**2,162125**	2,1174	2,09955	2,08065	**2,08325**	**2,08365**	2,09805	1,7661
PT08.S1(CO)	1145,712	1122,225	1143,801	**1145,712**	**1145,764**	1146,14	**1145,758**	1146,439	1146,66	1145,262	1137,308
NMHC(GT)	218,8118	99,997	135,835	**218,8118**	**218,874**	221,35	217,579	226,188	219,3265	209,036	118,3845
C6H6(GT)	10,23014	10,02042	10,10336	**10,23014**	**10,23303**	10,2128	10,20904	**10,23241**	10,25396	10,20619	10,02994
PT08.S2(NMHC)	947,1089	927,6931	942,3148	**947,1089**	**947,1348**	946,6791	**947,105**	**947,348**	948,1786	946,9324	936,6516
NOx(GT)	134,5636	107,449	115,7105	**134,5636**	**132,1225**	130,4725	**131,1715**	130,6725	130,649	**131,2995**	108,9335
PT08.S3(NOx)	963,8149	944,0567	959,0781	**963,8149**	**963,8025**	964,9787	961,9372	**963,5098**	962,7898	962,8911	953,3125
NO2(GT)	96,86788	77,349	96,8945	**96,86788**	**95,5415**	94,746	95,8735	94,68	94,7225	**95,626**	79,7555
PT08.S4(NO2)	1600,405	1567,597	1596,153	**1600,405**	**1600,378**	1598,812	1600,237	**1600,693**	1601,391	**1599,833**	1587,174
PT08.S5(O3)	982,497	**962,3558**	972,3752	**982,497**	**982,5315**	981,6596	983,3151	983,7048	984,5095	981,672	967,7473
T	17,51508	17,15603	**17,44661**	**17,51508**	**17,51418**	**17,50746**	17,49706	**17,50466**	**17,51219**	**17,5281**	17,28108
RH	48,10241	47,11631	47,65956	**48,10241**	48,0944	47,99715	48,12613	48,12957	**48,10872**	**48,0834**	47,3044
AH	0,908318	0,889697	0,897394	**0,908318**	**0,908168**	0,905682	**0,908701**	**0,908105**	**0,908174**	**0,908501**	0,897394

	ignore	special	common	mean	svd	knn	rf	lr	em	k-means	zet
CO(GT)	1,329	1,470	1,209	1,188	1,235	**1,305**	**1,313**	**1,317**	**1,325**	1,293	1,456
PT08.S1(CO)	228,551	278,452	226,581	226,196	**226,197**	**227,814**	**226,586**	**226,572**	**226,951**	**226,217**	233,540
NMHC(GT)	204,460	176,012	157,768	138,177	138,226	**189,591**	**193,547**	**189,436**	176,689	166,600	176,864
C6H6(GT)	7,112	7,187	**7,093**	7,039	7,039	7,072	7,040	7,041	**7,062**	7,041	**7,174**
PT08.S2(NMHC)	259,208	289,537	**258,669**	256,536	256,537	**257,962**	256,536	256,599	257,416	256,539	266,531
NOx(GT)	77,355	87,706	**78,656**	69,120	71,896	75,483	74,685	75,877	**76,227**	74,990	86,117
PT08.S3(NOx)	254,688	286,701	**254,181**	252,062	252,062	**254,512**	252,492	252,134	252,763	252,143	262,313
NO2(GT)	32,106	48,307	28,688	28,688	30,295	31,478	31,046	**31,962**	**32,015**	30,967	45,021
PT08.S4(NO2)	279,923	358,058	278,593	277,037	277,037	**279,626**	277,079	277,149	**277,916**	277,066	291,750
PT08.S5(O3)	364,202	386,414	367,179	360,448	360,448	**363,668**	**361,298**	360,804	**361,853**	360,493	374,597
T	5,402	5,895	5,367	5,346	**5,346**	5,363	5,348	**5,350**	5,349	5,347	5,586
RH	16,345	17,555	**16,464**	16,176	16,177	**16,316**	16,177	16,183	16,186	16,177	17,092
AH	0,201	0,237	0,213	0,199	0,199	**0,202**	**0,199**	**0,199**	**0,199**	**0,199**	0,213

Аналіз результатів

- Кращий результат показав алгоритм відновлення пропусків за допомогою випадкового лісу. Гарні результати показують методи, засновані на k найближчих сусідів, EM-алгоритмі і лінійної регресії.
- Заміна спеціальним значенням показала найгірший результат.

Дякую за увагу!

ДОДАТОК Б Лістинг програми

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn import preprocessing
import sys

import imputers
import random_deletion
import utils

_algs = ['RF', 'LR', 'kNN']
_methods = ['ignore', 'special', 'common', 'mean', 'svd', 'knn', 'rf', 'lr', 'em', 'k-means',
'zet']
_colors = {'zet': '#CC0000',
        'special': '#888800',
        'common': '#66CC00',
        'mean': '#00CCCC',
        'svd': '#0066CC',
        'knn': '#6600CC',
        'rf': '#CC00CC',
        'lr': '#333333',
        'em': '#FF8000',
        'k-means': '#CCCC00'}
_datasets = ['krkp', 'creditg', 'segment']

# DATA IMPUTING
# data_real, target, clf, cv, missing_frac_range, num_iter, sp_value, add_binary
target = None
```

```
clf = 0
cv = 0
missing_fraction = [0, 0.02, 0.04, 0.06, 0.08, 0.1, 0.15, 0.2]
num_iteration = 1
sp_value = 0
add_binary = 0

#CODE FOR SEGMENT
segment = pd.read_csv('dataset\segment.csv')
segment = segment.drop([0], axis=0).drop(['class'], axis=1)
segment_normalized = preprocessing.normalize(np.array(segment), axis=0)
segment_normalized = pd.DataFrame(segment_normalized)
segment_normalized.index = segment.index
segment_normalized.columns = segment.columns


acc, rmse_segment = utils.make_experiments(segment_normalized, target, clf, cv,
missing_fraction, num_iteration, sp_value, add_binary)
print(f'Accuracy:\n{acc}\nRMSE:\n{rmse_segment}')
rmse_segment.to_csv('SegmentNORM_rmse.csv')
rmse_segment.drop('ignore', axis=0)
indexes = rmse_segment.index
colmns = rmse_segment.columns

for method in indexes:
    plt.plot(colmns, rmse_segment.loc[method, :], label=method)
plt.xlabel("Missing value fraction", fontsize=12)
plt.ylabel("RMSE", fontsize=12)
```

```
plt.legend(loc='center left', bbox_to_anchor=(1, 0.5), fontsize=10)
plt.show()
        import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import sys

import imputers
import random_deletion
import utils

_algs = ['RF', 'LR', 'kNN']
_methods = ['ignore', 'special', 'common', 'mean', 'svd', 'knn', 'rf', 'lr', 'em', 'k-means',
'zet']
_colors = {'zet': '#CC0000','special': '#888800','common': '#66CC00','mean':
'#00CCCC',
'svd': '#0066CC','knn': '#6600CC','rf': '#CC00CC','lr': '#333333','em': '#FF8000',
         'k-means': '#CCCC00'}
_datasets = ['krkp', 'creditg', 'segment']

# DATA IMPUTING
# data_real, target, clf, cv, missing_frac_range, num_iter, sp_value, add_binary
target = None
clf = 0
cv = 0
missing_fraction = [0]
num_iteration = 1
sp_value = 0
add_binary = 0
```

```
# CODE FOR AIR QUALITY
# prepare data
air_quality = pd.read_excel('dataset\AirQualityUCI.xlsx')
air_quality = air_quality.drop(['Date', 'Time'], axis=1)
air_quality = air_quality.drop(index=air_quality.index[2000:], axis=0)
missing_mask = np.array(air_quality == -200)
air_quality[missing_mask] = np.nan
missing = utils.calc_missing_per(air_quality)

mean, dispersion = utils.make_experiments(air_quality, target, clf, cv,
missing_fraction, num_iteration, sp_value,
                                    add_binary, natural=True)
mean.columns = _methods
dispersion.columns = _methods
print(f'Mean:\n{mean}\nDispersion:\n{dispersion}')
mean.to_excel('results\AirQuality_MEAN.xlsx')
dispersion.to_excel('results\AirQuality_DISP.xlsx')
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import sys
from sklearn import preprocessing

import imputers
import random_deletion
import utils

_algs = ['RF', 'LR', 'kNN']
```

```
_methods = ['ignore', 'special', 'common', 'mean', 'svd', 'knn', 'rf', 'lr', 'em', 'k-means',
'zet']
_colors = {'zet': '#CC0000','special': '#888800','common': '#66CC00','mean':
'#00CCCC',
'svd': '#0066CC','knn': '#6600CC','rf': '#CC00CC','lr': '#333333','em': '#FF8000',
        'k-means': '#CCCC00'}
_datasets = ['krkp', 'creditg', 'segment']

# DATA IMPUTING
# data_real, target, clf, cv, missing_frac_range, num_iter, sp_value, add_binary
target = None
clf = 0
cv = 0
missing_fraction = [0, 0.02, 0.04, 0.06, 0.08, 0.1, 0.15, 0.2]
num_iteration = 1
sp_value = 0
add_binary = 0

# CODE FOR CANCER
# preparing data
cancer_analyze = pd.read_excel('dataset\BreastTissue.xlsx')
cancer_analyze_normalized = preprocessing.normalize(np.array(cancer_analyze),
axis=0)
cancer_analyze_normalized = pd.DataFrame(cancer_analyze_normalized)
cancer_analyze_normalized.index = cancer_analyze.index
cancer_analyze_normalized.columns = cancer_analyze.columns



acc, rmse_cancer = utils.make_experiments(cancer_analyze_normalized, target, clf,
```

```
cv, missing_fraction, num_iteration, sp_value, add_binary)
print(f'Accuracy:\n{acc}\nRMSE:\n{rmse_cancer}')
rmse_cancer.to_csv('Cancer_rmse.csv')
rmse_cancer.drop('ignore', axis=0)
indexes = rmse_cancer.index
columns = rmse_cancer.columns

for method in indexes:
    plt.plot(columns, rmse_cancer.loc[method, :], label=method)
plt.xlabel("Missing value fraction", fontsize=12)
plt.ylabel("RMSE", fontsize=12)
plt.legend(loc='center left', bbox_to_anchor=(1, 0.5), fontsize=10)
plt.show()
        import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import sys
import time

import imputers
import random_deletion
import utils

_algs = ['RF', 'LR', 'kNN']
_methods = ['ignore', 'special', 'common', 'mean', 'svd', 'knn', 'rf', 'lr', 'em', 'k-means',
'zet']
_colors = {'zet': '#CC0000','special': '#888800','common': '#66CC00','mean':
'#00CCCC',
'svd': '#0066CC','knn': '#6600CC','rf': '#CC00CC','lr': '#333333','em': '#FF8000',
```

```
        'k-means': '#CCCC00'}
_datasets = ['krkp', 'creditg', 'segment']

rmse_segment = pd.read_csv('Segment_rmse.csv', index_col=0)
rmse_segment = rmse_segment.drop('ignore', axis=0)
indexes = rmse_segment.index
colmns = rmse_segment.columns

for method in indexes:
    plt.plot(colmns, rmse_segment.loc[method, :], label=method)
plt.xlabel("Missing value fraction", fontsize=12)
plt.ylabel("RMSE", fontsize=12)
plt.legend(loc='center left', bbox_to_anchor=(1, 0.5), fontsize=10)
plt.show()
```